Improving ensemble price prediction in multi-agent market places to guide strategic sales decisions

Erasmus School of Economics

Master Economics & Informatics

Computational Economics Specialization

Authors:

R.J. Romke de Vries, Bsc Student-id 277260
O.B ter Haar, Bsc Student-id 290074

Supervisors: Prof. dr. ir. Uzay Kaymak and dr.Wolf Ketter

Rotterdam, 1 April 2010

Abstract

This research focuses on improving ensemble price prediction in multi-agent market places to guide strategic sales decisions. The ensemble price prediction mechanism is implemented in the MinneTAC agent. The MinneTAC agent is an agent-based-computer model developed by the University of Minnesota in cooperation with the Erasmus University. The MinneTAC agent is competing in the Trading Agent Competition for Supply Chain Management (TAC SCM). The TAC SCM competition was developed to come to the best solution of an agent-based-computer model capable of handling a dynamic supply chain. The main goal of this research is to improve the ensemble price prediction mechanism of the MinneTAC agent with techniques like bootstrapping and dynamic economic regime transition probabilities.

Acknowledgements

We are Professor Uzay Kaymak and Wolfgang Ketter highly indebted for their supervision and support in realizing this master thesis. Thanks to the good supervision, we had a steep learning curve.

We would like to thank John Collins, from the University of Minnesota, for this very helpful technical assistance and knowledge about the software architecture of the MinneTAC agent.

Contents

1 Introduction **4**
- 1.1 A brief overview of the problem . . . 5
- 1.2 The research questions and the academic value . . . 6
  - 1.2.1 The research related to improving ensemble price prediction . . . 6
  - 1.2.2 Fuzzy economic regime transitions . . . 8
  - 1.2.3 The academic value of the research . . . 9
- 1.3 Methodology . . . 10
- 1.4 Outline . . . 10

2 Price prediction **12**
- 2.1 Ensemble price prediction . . . 12
- 2.2 Economic regime price prediction . . . 14

3 A description of the TAC SCM game and the MinneTAC agent **17**
- 3.1 The Trading Agent Competition for Supply Chain Management 17
  - 3.1.1 The TAC SCM game scenario . . . 18
  - 3.1.2 The working of the TAC SCM game . . . 19
- 3.2 The software architecture of the MinneTAC agent . . . 22
- 3.3 Price prediction in TAC SCM agents . . . 25

4 The ensemble price prediction mechanism of the MinneTAC agent **29**
- 4.1 An overview of the price prediction mechanism of the MinneTAC agent . . . 29
- 4.2 The identification of the economic regimes . . . 32
- 4.3 How the game data is processed online . . . 35
- 4.4 The ensemble price prediction mechanism of the MinneTAC agent . . . 36
  - 4.4.1 The price prediction methods . . . 36
  - 4.4.2 The dynamic weighting mechanism . . . 39
- 4.5 Summary . . . 41

5 The improvements in the ensemble price prediction mechanism of the MinneTAC agent 42
5.1 The experimental design 42
5.1.1 The required number of experiments 44
5.1.2 How the game data is obtained and analyzed 45
5.1.3 The performance measures 46
5.2 The methodology to improve ensemble price prediction 47
5.2.1 The optimal quantity of historical price predictions . . 47
5.2.2 The optimal weights for each price prediction 49

6 The outcomes of improving ensemble price prediction. 53
6.1 The optimal variances calculations 53
6.1.1 The exponential variance calculation method 53
6.1.2 The block weighting variance calculation method . . . 57
6.2 A discussion of the weights of each price predictor 61
6.3 The discussion of the bootstrapped results 65

7 Fuzzy economic regime transition probabilities in the MinneTAC agent 70
7.1 The different game phases during the TAC SCM competition 71
7.2 The Fuzzy economic regime transition mechanism 71
7.3 The experimental design and methodology 73
7.4 The results 74
7.4.1 Dynamic economic regime transitions with exponential variance calculation 75
7.4.2 Dynamic economic regime transitions with block weighting variance calculation 76
7.4.3 A comparison between block weighting and exponential variance calculation 77

8 Conclusion and future research 79
8.1 Improving ensemble price prediction 79
8.2 The implementation of dynamic economic regime transition probabilities 80
8.3 Vision of the authors 81

Appendix 87

Chapter 1

Introduction

Globalization and the pressure to reduce cost are influencing companies to further optimize their supply chains [1]. Like information on the internet, goods are moving around the world with greater efficiency [1]. Supply chain management is a set of approaches utilized to efficiently integrate suppliers, manufactures, warehouses and stores, so that merchandize is produced and distributed at the right quantities, to the right locations and at the right time, in order to minimize system-wide costs while satisfying service level agreements [2]. The supply chain ends when the finished products are delivered to the customer, through a distribution system that mostly includes an inventory system[3]. Companies have to manage these complex logistic networks.

The TAC SCM game was designed to come to the best solution for an agent based computer model that is capable of dealing with the problems of a flexible supply chain. The challenge of the TAC SCM game is that the participating agent-based computer models have to make fully autonomous decisions in a non-deterministic environment, that simulates a dynamic supply chain. Human intervention is not allowed during the game [4]. The TAC SCM competition is an environment to test automated decision support systems. When companies are able to implement such an automated decision support system, better decision making could improve their competitive position [4].

An agent based computer model is defined as a model implemented in a computer that is able to make fully automated decisions without human intervention. Flexibility is required to make fully automated decisions. Flexibility is defined as the capability of the model to be responsive, proactive, and social. Responsiveness means the ability to perceive the environment and to react to changes that occur in the environment. Proactive stands for the ability to exhibit opportunistic, goal-directed behavior and taking the initiative where appropriate. The term social stands for the ability to interact with other agents [5].

This research focuses on how to improve the price prediction mechanism

of the MinneTAC agent. The MinneTAC agent is an agent-based computer model, developed by the university of Minnesota in co-operation with the Erasmus university. The MinneTAC agent competes with other agent based computer models in the Trading Agent Competition for Supply Chain Management (TAC SCM) [6]. Besides winning the TAC SCM competition, the MinneTAC research group is interested in implementing the techniques applied in the MinneTAC agent in real world applications. Important research topics include economic modelling under uncertainty in environments where the actions of other agents have a significant impact and where the actions of the agents are only partly observable. To support the project, research in software engineering is applied to support a dynamic research agenda [6].

1.1 A brief overview of the problem

In the Trading Agent Competition for Supply Chain Management, Agent based computer models have to compete with each other in a complex resource allocation situation. In the game, the market conditions are constantly changing. The agent has to take this into account. In the TAC SCM game, Agent based computer models compete in a dynamic supply chain setting. In the TAC SCM game, Artificial intelligence techniques are added to the Agent based computer models. These techniques are used to enable optimal decision making during the games. Figure 1.1 gives an overview of the challenges of the TAC SCM game.

In the game scenario, there is only limited information available about the state of the markets and the strategies of the competing agents. Based on this limited information, the agent has to make decisions about resource allocation and the pricing of the products produced. At the same time, the agent has to compete with other agents in procurement of parts and in the sales of finished products [4].

The goal of the TAC SCM game is to capture the challenges involved in supporting dynamic supply chain practices [7]. In the game, it is essential that the future market prices are correctly predicted. The agent which is able to predict the future market prices better than its competitors will win the game. By modeling the interactions that take place in a dynamic supply chain, better insight is obtained [4]. These findings could be implemented in an industrial environment.

The winner of the TAC SCM game is the agent that ends up with the highest amount of money in its bank account. In the game, the agents have to produce personal computers (PCs) and sell these produced PCs for the highest possible price. The agent has five other competitors in the game. Together with its competitors, the agent has to procure parts and sells its produced PCs. In figure 1.1, an overview of the interactions in a TAC SCM game is given. At the procurement stage, the agent has to bid for parts, and the agent with the best bid obtains the parts. When the agent sells

the produced PCs, the agent has to estimate the market demand to price its PCs accordingly. The problem is that the market demand and prices constantly change, which makes price prediction important.

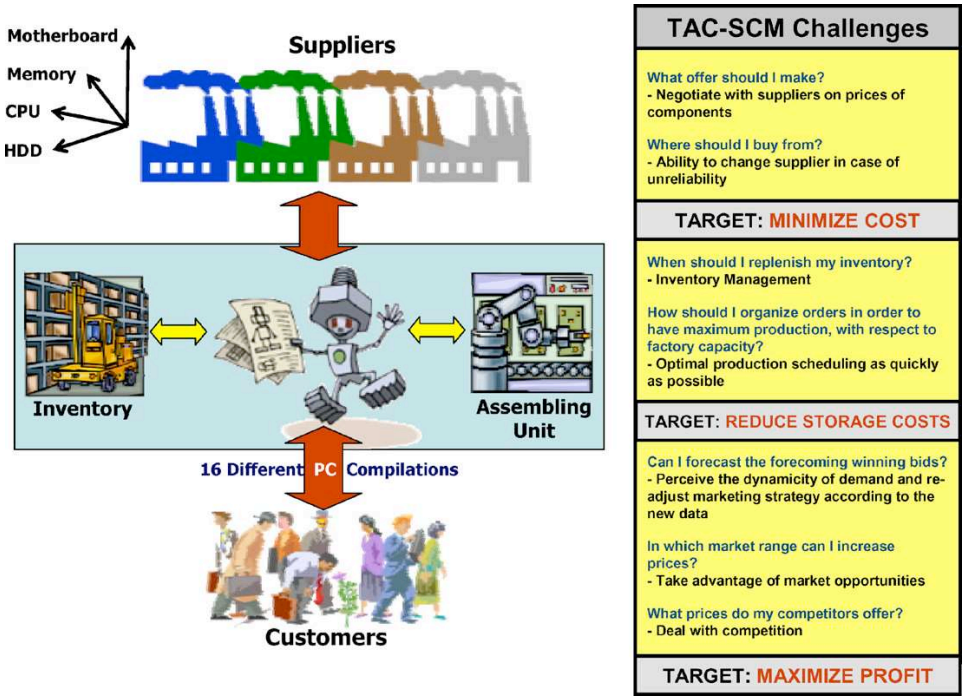

Figure 1.1: A summary of the TAC SCM competition [8]

1.2 The research questions and the academic value

This research tries to improve the price prediction mechanism of the MinneTAC agent in two ways. In the first place, the concept of ensemble price prediction is improved. Secondly, fuzzy economic regime transition probabilities are implemented in the price prediction mechanism of the MinneTAC agent.

1.2.1 The research related to improving ensemble price prediction

Ensemble price prediction is a methodology that combines multiple price prediction methods. Ensemble price prediction can lead to a better performance because the disadvantages of the price predictors in the ensemble are compensated by taking a weighted average price [9]. A weighting mechanism takes an weighted average price based on the predictive performance of each price prediction method. Therefore, the weighting mechanism is an important part in the ensemble price prediction mechanism. In this research, the weight for each price prediction method is based on the inverse of the variance in past price predictions [9]. The variance measures how well

the price prediction mechanism is able to predict the future market prices [9].

This research is divided in four research questions. Here below, the research questions are explained. The methodology is explained in detail in chapter five.

What is the optimal quantity of historical price predictions to calculate the weight of each price prediction method as accurate as possible?

The ensemble price prediction mechanism of the MinneTAC agent combines the price predictions of each price prediction method. A dynamic weighting mechanism calculates the weight of each price prediction in the ensemble. The weight of each price prediction method is based on the variance in previously predicted prices. The variance is based on the squared difference between the actual prices in the game and the prices that are predicted by the MinneTAC agent. How fast should the weights change to calculate the future prices as accurately as possible? The MinneTAC agent can use a block weighting or an exponential mechanism to calculate the variance. The block weight horizon is the number of past price predictions to calculate the variance of each price prediction method. With a small block weight horizon, the weight of each price prediction method is more reactive to changes in the variance. In the exponential weighting mechanism, the smoothing parameter τ, determines the influence of past price predictions. τ is pronounced as tau. A small τ value means that the variance is more reactive to changes in the predictive performance of each price prediction method. Hence, exponential smoothing applies a set of declining weighs to all past data and the τ determines the speed of the decline. Experiments are executed to determine the optimal values for the block weight horizon and τ value.

What is the optimal weight of each price prediction method in the ensemble?

In this research question, the optimal weight of each price prediction method in the ensemble is determined. The goal of ensemble price prediction is to compensate the shortcomings of each price prediction method. A dynamic weighting mechanism has to determine the optimal weight for each price prediction method for price predictions up to 20 days ahead in the future. The dynamic weighting mechanism is called dynamic because the weights of every price predictor are constantly updated based on the predictive performance of each price prediction method. This research tries to find the optimal weights for each price prediction method during a TAC SCM game. When the optimal weights are known, it can be determined if the ensemble consists of a optimal mix of price prediction methods. The goal of ensemble price prediction is that the shortcomings of each price prediction method are compensated by taking a weighted average price. Is this achieved in the MinneTAC agent? An understanding of the behaviour of each price prediction method is acquired by calculating the weight of each price prediction for every virtual game day.

Did the ensemble price prediction mechanism outperform the individual price prediction methods?

Ensemble price prediction is only effective if the weighted average ensemble price is more accurate then the prices predicted by each individual price prediction method. When the ensemble price is more accurate then the individually predicted prices, the shortcomings of each price prediction method are compensated. Is this happening in the MinneTAC agent?

What is the influence of bootstrapping on the performance of the MinneTAC agent?

The MinneTAC agent has a dynamic learning mechanism that learns the optimal weight of each price prediction method in the ensemble. When the game starts, each price prediction method has an equal weight in the ensemble. This does not optimally compensate the shortcomings of each price prediction method. At the start of each TAC SCM game, the MinneTAC agent has to learn the optimal weights. When the MinneTAC agent could start with the optimal weights, a costly learning process is avoided. This is called bootstrapping. In previously held TAC SCM competitions, the MinneTAC agent under performed in the first phase of the game. With bootstrapping, this under performance is avoided. This could significantly improve the score of the MinneTAC agent in future TAC SCM competitions. Experiments are performed to test the gain in the predictive accuracy of the MinneTAC agent and to test if it is better to bootstrap the agent with the optimal weight for the first phase of the game or to use the average weight taken over the complete duration of the game. In the end, the gain in the predictive performance of bootstrapping versus non-bootstrapping is determined to establish the added value of bootstrapping.

1.2.2 Fuzzy economic regime transitions

Economic regimes are the foundation of the price prediction mechanism of the MinneTAC agent. The intuition behind economic regimes is that prices reflect future expectations in the demand and supply relationship. Research has found that the relationship between demand and supply is not constant during a TAC SCM game. Currently, the MinneTAC agent is implemented with on set of regime transition probabilities. To improve the price prediction mechanism of the MinneTAC agent, for every phase of the game, customized economic regime transition probabilities are implemented. Research has identified three different game phases. In the beginning of a TAC SCM game, there is a shortage of components that results in higher market prices for components. After approximately 30 days, this shortage is resolved. Component prices reach a normal level. At the end of the game, the participating agents start dumping inventories. This results in significant lower market prices.

The customized regime transition probabilities can lead to an increase in the predictive performance of the MinneTAC agent. A Fuzzy weighting mechanism is implemented to enable a smooth transition between the different game phases. A Fuzzy weighting mechanism is used because Fuzzy weighting enables a gradual change between the different economic regime transition probabilities [10]. The MinneTAC agent makes strategic and tactical decisions and changing the economic regime transition probabilities to abruptly would embroil the strategic decisions of the MinneTAC agent. The Fuzzy weighting mechanism enables a smooth transition without disturbing the strategic decisions.

1.2.3 The academic value of the research

The MinneTAC agent uses ensemble price prediction to make accurate future price predictions. The idea of ensemble price prediction is that the shortcomings of each price prediction method are compensated by taking an weighted average price. The shortcomings of each price prediction method could only be compensated when each price prediction method performs optimal in a different time space. This research investigates if ensemble price prediction is more accurate than the use of a single price prediction method. Is ensemble price prediction able to give a significant increase in the predictive performance? When ensemble price prediction results in a significant gain in the predictive performance, this could be implemented in real world applications.

Secondly, research is performed on the concept of bootstrapping. Bootstrapping is a form of supervised learning in which training data is used to avoid a learning process [11]. In the MinneTAC agent, bootstrapping has to avoid the learning of the weights of each price prediction method in the ensemble. How costly was the learning process of the optimal weights of each price prediction method in the MinneTAC agent? Did bootstrapping lead to a significant increase in the predictive performance of the MinneTAC agent in the first phase of a TAC SCM game?

In the third place, Fuzzy economic regime transitions are implemented to enable a smooth transition between the economic regime transition probabilities of each game phase. This research investigates the importance of customized economic regime transition probabilities for each game phase and the effects of a gradually changing the customized economic regime transition probabilities.

All the methods mentioned above could also be implemented in decision support systems of real life applications. If the improvements are successful in the MinneTAC agent, success could also be achieved in the real world.

1.3 Methodology

This section gives a brief overview of the employed methodology. Empirical research has to be performed to analyze the results from experiments with the MinneTAC agent. There is started with a literature study to get a good perspective about the background of the research. The literature study evaluates price ensemble price prediction and price prediction based on economic regimes.

The research questions related to ensemble price prediction and improving the Markov predictor are answered by carrying out experiments. These experiments simulate TAC SCM games. In these games, the MinneTAC has to compete with its rivals. Since the MinneTAC is one of the leading agents in the TAC SCM competition, the competitors are agents that showed the best performance during the recent TAC SCM competitions. This provides a good benchmark for the performance of the MinneTAC agent. The university of Minnesota developed a special game environment to perform the experiments. In this environment, the TAC SCM games are simulated and it is possible to analyze the game data.

In the experiments, game data is logged on the game servers of the university of Minnesota. This game data is used to determine the pricing that occurred during the game. The MinneTAC also logs important game data. From the log files of the MinneTAC agent, the predicted prices are extracted. This enables a comparison that indicates how well the MinneTAC is able to predict the future market prices accurately. The Root mean square error is used to indicate the predictive performance of the MinneTAC agent. The root mean square error is a widely used method to determine the predictive accuracy.

1.4 Outline

This section gives an overview of the content of the research that follows hereafter. Here below, the content of each chapter is summarized.

Chapter 2 starts with a literature review. In the literature review, an overview is given of related research. Topics like ensemble price prediction, price prediction with economic regimes and price prediction by other TAC SCM agents are evaluated.

Chapter 3 explains the working of the TAC SCM game and the software architecture of the MinneTAC agent.

Chapter 4 explains the working of the price prediction mechanism of the MinneTAC agent. The price prediction mechanism is composed of multiple price prediction methods and a dynamic weighting mechanism. This chapter also explains how the future market prices are used for tactical and strategic

decision making.

Chapter 5 explains the methodology and experimental design the answer the research questions related to improving ensemble price prediction with block weighting variance error calculation.

In Chapter 6, the research questions related to improving price prediction with Fuzzy regime transition probabilities are explained.

Chapter 7 evaluates the possibilities for future research and gives the conclusion.

Chapter 2

Price prediction

This chapter gives an overview of the related literature of ensemble and economic regime price prediction. This research tries to improve the price prediction mechanism of the MinneTAC agent. The MinneTAC agent uses 'economic regime' and 'ensemble price' prediction to make future price predictions as accurately as possible. This chapter explains economic regime price prediction ensemble price predictions and gives and overview of the related literature.

2.1 Ensemble price prediction

The main idea behind ensemble methods is that diverse perspectives on different aspects could be combined to provide a high quality solution. Ensemble prediction aims to create a prediction or classification mechanism that is more robust than a single method [12]. In the literature, there are three methods proposed to compose an ensemble. These methods are Bagging, Boosting and Stacking [13] [12].

Bagging creates an ensemble by using the same classification or prediction methods for randomly generated training data [14]. Bagging trains each prediction or classification separately and combines the results with a weighted voting mechanism [14].

Boosting uses the same prediction or classification method to make predictions or classifications. The difference between Boosting and Bagging is that Bagging trains each method independently while Boosting trains each method dependently. Each prediction or classification is influenced by the previous method. This is performed by enforcing a new method to train each incorrectly handled input sample of the previous method [15].

The Stacking ensemble prediction approach is based on different prediction methods. The method inherits the advantages but also suffers of the disadvantages from each individual prediction or classification method [12].

Therefore, the performance of this method is not always better than an individual prediction method [12]. Zou Wan et al. propose a concept that selects suitable prediction or classification methods from a pool that performs more effective then combining all classifiers simultaneously [16].

Cihli Hung and Jing-Hong Chen have implemented an ensemble classification mechanism to predict bankruptcy [13]. The ensemble consisted of a decision tree, a back propagation neural network and a support vector machine. Based on the expected probabilities of both bankruptcy and non-bankruptcy, this ensemble provided an approach which inherited the advantages and compensated the disadvantages of the classification techniques. A meta-learner, which selects suitable classifiers based on an expected probability of each individual classifier to replace the voting policy was used to combine the individual classification methods. Consistently, the ensemble performed better than other weighting or voting systems for bankruptcy prediction [13].

Jau-Jia Guo and Peter Luh improved market Clearing Price Prediction by using an ensemble of neural networks [17]. The most important reason to use neural networks is that they are capable of inferring hidden relationships in data [11]. The problem is that a single neural network often misrepresents parts of the non-linear relationship in the data [11]. By using an ensemble of neural networks, this misrepresentation of the input-output relationship in the data is avoided. A weighting mechanism was used to combine the predictions of each individual neural network. The weights for each neural network are the probabilities that individual networks capture the true input-output relationship at that prediction instantly. This ensemble method performed better than individual neural networks and ensembles using weighted averaging methods [17].

Paulo J.L. Adeodata et all developed an ensemble of neural networks to forecast multiple time-series [18]. The results showed that the predictive performance of the ensemble degraded less then any single multilayer perceptron neural network [18].

Loris Nanni and Alessandra Lummi performed experiments with ensemble classifiers for bankruptcy prediction and credit scoring [19]. The results showed that the performance of the ensemble outperformed stand-alone classifiers [19]. The Random Subspace method was used to construct the ensemble. With this method, each stand-alone classifier used only a subset of all features for training and testing [20]. This method outperformed ensemble methods like bagging [19].

Hong-Hee Wan et al. implemented an ensemble approach to predict transcription start sites in human genomic DNA sequences [21]. Decoding of human genomic sequences has been a difficult practise. The annotation of enumerating DNA sequences is almost impossible without the support of computational techniques [21]. To improve the predictive accuracy, an

ensemble approach was implemented. The ensemble was build out of different neural networks. A majority voting, a weighted voting and Bayesian approach were used to combine the individual predictors. The ensemble performed better then the currently available single prediction methods [21].

Yong Seog Kim provided insight on the advantages and disadvantages of ensembles based on sampling and feature selection for database marketing [22]. The results showed that ensembles significantly improve the predictive performance of single classifiers at the cost of interpretability and computational resources. Especially predictors with high variances like neural networks and tree learners make a strong ensemble. Classifiers utilizing prior class distributions like support vector machines and naive Baysian classifiers only marginally benefit from ensembles [22]. For data sets with skewed distributions, ensemble predictors become useless [22].

2.2 Economic regime price prediction

Over the last 20 years, research is performed in forecasting methods based on the characteristics of the business cycles. A business cycle is defined as a succession of periods of rapid growth or slowdown or a decline in which the output fluctuates around its long run trend [23]. A business cycle is also called an economic regime [23]. J.D. Hamilton started with the development of price predictions based on economic regimes. Hamilton developed a model that was based on the idea that growth in the US gross domestic product could be captured through a dynamic model that switches between two regimes with different growth characteristics. The model had to reveal the most likely regime for each predicted sample and was based on transition probabilities. The transition probabilities are the probabilities of the new regime given the current regime. Markov transition matrices were used to determine the next regime [24].

The paper 'Business-cycle phases and their transitional dynamics', written by A.J. Filardo and J. Andrews extended the work done by J.D. Hamilton [25]. In this research, the differences between expansion and contraction phases of the business cycle were examined. This was performed with the switching method of J.D. Hamilton which take into account the incorporate time-varying probabilities of transitions between the phases. The marginal benefit of the time-varying transition probability, Markov switching mode, are also highlighted. This technique showed the high correlation between the evolution of the regime phases inferred from the model and the traditional reference cycles for monthly output data [25]. Many of the economic variables that determine the time-varying probabilities are helping to predict the turning points [25].

Wolfgang Ketter at all developed a prediction model based on economic regimes to predict future market prices. Market conditions were characterized as statistical patterns in historical market data. These patterns were

used to classify a range of economic environments that are expected in the future. A Gaussian Mixture Model was used to calculate the price density function and to determine the economic regimes. The economic regime model was tested in the MinneTAC agent [26]. The MinneTAC agent is an agent-based computer model, developed by the University of Minnesota and competes in the Trading Agent Competition for Supply Chain Management. The MinneTAC agent is one of the best performing agents.

The semiconductor industry has been one of the fastest growing industries in the past 20 years [27]. The semiconductor industry is also cyclical and demand patterns change rapidly [27]. This makes it difficult to predict turning points of business cycles. To deal with the cyclical behavior of the semiconductor industry, Wen-Hsum Lu and Yih-Luan Chyi developed a Markov Regime switching representing expansion and contracting of the semiconductor industry. Empirical results showed that the regime model successfully predicts the turning points in demand patterns. Data from January 1990 until March 2003 was used to test the model [27].

R. Ahrens, of Deka Investments GmbH, developed a regime model to predict recessions with interest rate spreads [28]. This research used Markov-regime switching models to evaluate the informal content of the interest rate term structure as a predictor of recessions in eight OECD countries. The empirical results suggested that the term spread is modelled as a two-stage regime switching process for all countries. The main advantage of this technique is that the lead times for forecasting discrete events like the ending of recessions are determined endogenously [28]. This implied, that compared to regression based techniques, the optimal forecasting horizon was free. The term spread of interest rates proved to be a good predictor. However, the results showed that the Markov regime switching probability as explanatory variable, did not significantly improved the forecasting ability of interest rate spreads [28]. This implied that there is a trade-off between sharp probabilities calculated by the regime model and accuracy of fitting independent recession data. The results showed that forecasters who prefer unambiguous signals could use this model [28].

For investors, the asset allocation decisions are a key determinant of their portfolio performance [29]. Massim Guidolin and Allain Timmermans developed a regime-switching model for stock and bond returns [29]. Four regime states were used in the regime-switching model. These were 'crash', 'slow growth', 'bull market' and 'the recovery state'. The regime states had different investment opportunities. Therefore, asset allocation varied significantly over time as the regime states changed and investors reversed their belief about the underlying regime state probabilities [29]. The conclusion of the research was that optimal asset allocation varies significantly across the regimes as the weights of the various asset types were strongly dependent on which state the economy was. Therefore, asset allocation varied significantly in the absence of outside predictor values such as dividend yield [29].

Timotheos Angelides and Nikolaos Tessaromafis developed a regime switching approach to determine the risk of price changes due to unique circumstances of a specific security, as opposed to the overall market risk [30]. This risk is called the idiosyncratic risk [30]. The evidence on the inter-temporal relation between idiosyncratic risk and future stock returns is conflicting and confusing. A first order Markov process modelled the regime transition probabilities. A Gaussian distribution was used to determine the transition probabilities. The regime model allowed the conditional distribution of stock returns to vary across two volatility states. The evidence suggested that in the US, idiosyncratic risk is a predictor of future stock prices only during the low variance regime state. In the high volatility regime state, the relation between idiosyncratic risk and stock returns is not statistically significant [30].

Timothy D. Mount et all predicted price spikes in electricity markets using a regime-switching model with time-varying parameters [31]. The research showed that a stochastic regime-switching model with time-varying parameters can capture the type of volatile priced behavior observed in many deregulated spot markets for electricity. The mean prices are divided in two regimes and the transaction probabilities are specified as functions of the offered reserve margin and the system load. Markov-switching in the model allowed the high price regime to be more persistent than in a simple binomial jump process [31]. The regime model replicated the observed price volatility well. This implied that the regime model is potentially useful for evaluating forward contracts and investment decisions in energy markets[31].

Chapter 3

A description of the TAC SCM game and the MinneTAC agent

The MinneTAC agent successfully competes in the Trading Agent Competition for Supply Chain Management (TAC SCM). This chapter describes the TAC SCM game, the software architecture of the MinneTAC agent and the future price predictions systems of competing TAC SCM agents. This chapter is divided in three sections. The first part of this chapter evaluates the TAC SCM game. The second part of this chapter explains the software architecture of the MinneTAC agent. The third part of this chapter gives an overview of the price prediction systems of competing TAC SCM agent.

3.1 The Trading Agent Competition for Supply Chain Management

The Trading Agent Competition is an international forum with the goal to promote and encourage highly qualitative research agent-based computer models in complex dynamic environments [32]. The first Trading Agent Competition was organized in 2000. Since 2003, the Trading Agent Competition was extended wit a supply chain management game scenario and the initial game scenario was called TAC Classic. The supply chain management scenario was called the Trading Agent Competition for Supply Chain Management (TAC SCM) [4]. The TAC SCM game scenario and the working of the TAC SCM game are described in the next sections. Below is a brief overview of the TAC Classic game.

In the TAC Classic game scenario, each agent-based computer model is a travel agency that has to compose travel packages. The agent is acting on behalf of its customers. Every agent has eight customers. The agent has to take specific preferences of its clients into account. The primary objective of the agent is to maximize the satisfaction of its customers. The agent that

has the most satisfied customers wins the game. Human intervention is not allowed during the game [32].

3.1.1 The TAC SCM game scenario

The TAC SCM game was developed by a team of researchers from the e-Supply Chain Management Lab at Carnegie Mellon University, the University of Minnesota, and the Swedish Institute of Computer Science (SICS) [4]. The TAC SCM game was developed to deal with the implementation of dynamic supply chain management practices [4]. Today's supply chains are getting more and more dynamic to match market demands in a globalizing economy. The globalizing economy and the increase in customer expectations regarding costs and services have forced manufacturers to improve processes within their supply chains [33]. The implementation of dynamic supply chain management has proven to be a difficult and highly complex procedure. The TAC SCM competition provides an ideal testing environment for decision support systems that are able to facilitate more dynamic supply chains. A dynamic supply chain is characterized by short term contracts between different contractors. The contracts are constantly re-negotiated to get better prices and to match market demand without building up large inventories. This compared to today's more static supply chains, that depend on long term relations among key trading partners [4].

A TAC SCM game consists of 220 virtual game days in which six agent-based computer models have to produce personal computers (PCs). In the first place, the agents have to compute for customer orders. The agent that offers the best price gets the deal. In the second place, the agents have to procure parts to produce the PCs. In this case, the agents have to compete to get the best prices from the suppliers. When the game starts, the participating agents have no inventories and no money in their bank account. To buy inventories, money has to be lent by the bank. The agents can have a negative balance on their bank account. Interest has to be paid over the money borrowed at the bank. During the game, agents have to make as much profit as possible. The agent with the highest amount of money in its the bank account is declared the winner at the end of the game. In figure 3.1, the TAC SCM game interactions between the agents, suppliers and customers are displayed [4].

The TAC SCM game presents a broad range of supply chain situations. At the end of the game, the agent with the highest amount of money on its bank account is declared the winner [4]. The TAC SCM game has a complexity that prohibits any realistic attempt to win the game from a game theoretic perspective. This implies that it is not possible for strategies that can be described as 'in equilibrium' in any way will to emerge. Thus a successful TAC SCM agent has to be robust to a wide variety of opponents and different game situations. Ideally, a good TAC SCM agent has to be adaptive to the environment and to the strategies of its competitors [32].

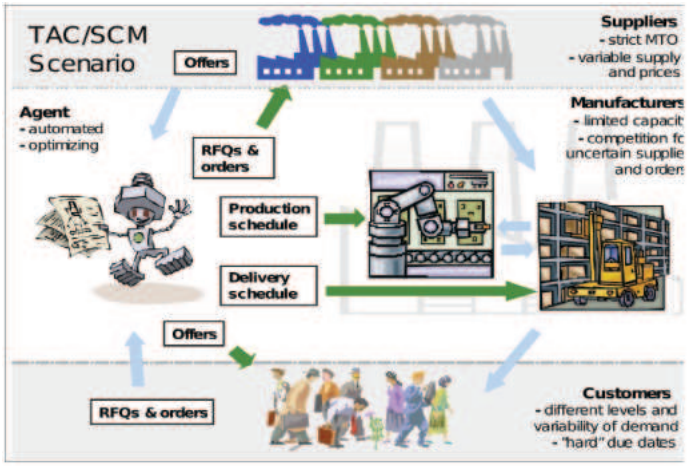

Figure 3.1: An overview of the tasks in the TAC SCM game [4].

The next sections explains the working of the TAC SCM game.

3.1.2 The working of the TAC SCM game

This section explains the working of the TAC SCM game. During a TAC SCM game, each agent has four tasks. These are the negotiation with suppliers, bidding for customer orders, managing the daily production and assembly of PCs in the factory and shipping the finished PCs to the customers. These tasks have to be performed every virtual game day without human intervention.

The game-server is responsible for customer and the supplier simulation. The game-server also manages the bank accounts, the factory and the warehousing for all the participating agents [4]. These functions do not have to be implemented in the agents software. The TAC SCM agent-ware is a software package that enables a smooth communication between the agent and the game-server. The packages have to be implemented in every agent. The game-server saves all important information generated during the game. This information could be used to analyze the performance of the agent after the game is completed [4].

The participating agents communicate with the game server using messages. These messages contain important information about the state of the game. At the beginning of each game day, the agent receives messages from suppliers, customers, the bank and the factory [4].

The messages from the suppliers contain offers for components in response to request for quotes. A request for quotation is a standard business process with the purpose to invite suppliers into bidding on specific products or services. Secondly, these messages state the supplies to satisfy previously made orders. The supplies are used for the production in the next day [4].

The messages from the bank have a statement about the amount of money in the agents bank account.

The factory sends a message that contains an inventory report and the number of finished PCs that are ready to be shipped to customers [4].

During every game day, every agent has to make five important decisions.

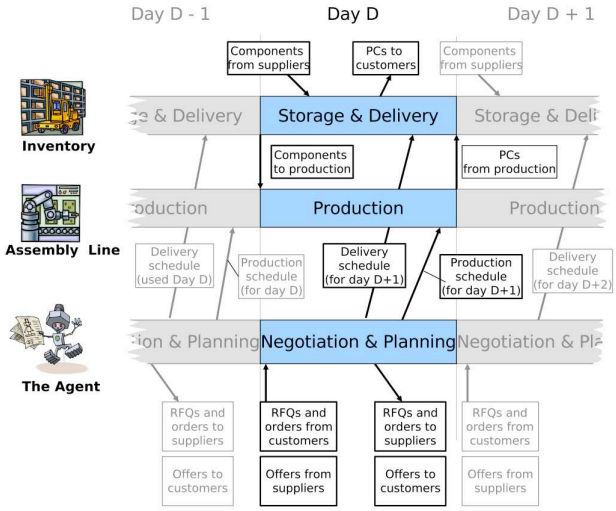

Figure 3.2: An overview of the major decisions that every TAC SCM agent faces during the game [4].

The five major decisions during the TAC SCM game

1. The agent has to decide how to bid on the request for quotes sent by potential customers.

2. The agent has to procure new components by sending request for quotes to the suppliers.

3. Suppliers send offers to the request for quotes sent by the agent the previous day. The agent has to decides which request for quotes to accept.

4. The agent has to allocate the components

5. The agent has to decide which finished PCs to ship to the customers. The customers are able to send penalties for late deliveries. This could be specified in the request for quotation. Therefore it is important that the agent delivers the finished PCs on time.

Before the end of the day, the agent has to make the above mentioned decisions. At the end of the day, the agent needs to respond to the customers

and suppliers. The decisions are communicate by messages send to the game server [4].

In the TAC SCM competition, the suppliers keep track of the reputation of the agents. The suppliers prefer to do business with reliable and loyal agents. The customers are also offering better prices to agents that have a good reputation. Every supplier measures the reputation of all the agents in the game. The reputation is based on the quantity of purchased products divided by the total number of request for quotes send by the agent [4].

Below is explained how the agents customers, suppliers, the bank account, daily production and inventory management are modeled in the TAC SCM game.

The customers

The customers request PCs during the game. The customer demand has a random distribution. The customers can buy sixteen different types of PC in three market segments during the game. In the game, the customers send request for quotes to the participating agents. In the request for quotes, the number of PCs, the reserve price and the due date are specified. Customers take offers from Agents, that are able to deliver the requested quantity of PCs before the due date with a price that is a least equal to the reserve price, in consideration. The agent that offers the best price gets the offer. This agent has to deliver the order before the due date. When the agent can not deliver the order before the due date, the customer sends a financial penalty to the agent. The penalty is specified in the request for quotation. A finished PC has to be shipped at least one day before the due day date in order to arrive on time [4].

The suppliers

In a TAC SCM game, there are eight distinct suppliers. There are two suppliers for each component type. Each supplier produces multiple types of each component. During the game, the suppliers operate an a make to order basis. The suppliers are revenue maximizing entities [4].

Each game day, the agents can place only a limited number of request for quotes. An agent may send five request for quotes for a component produced by a supplier. The agent may not send more then ten request for quotes to a supplier. A request for quotation consists of a specified quantity of a component, a maximum price the agent is willing to pay and a due date before the components have to be delivered to agent. At the end of each game day, the suppliers collect the request for quotes. The market prices for the components are determined based on the ratio between demand and supply. Each supplier tries to maximizes its revenue. The suppliers respond to the agents with offers at the beginning of the next game day. The agents

have to decide which offers to accept. When an agent accepts an offer, it has to send a message to the supplier at the end of game day [4].

The bank account function

The game-server manages the bank accounts for the agents. At the beginning of each virtual game day, the server sends messages to the agents with information over the amount they have in the bank account. At the beginning of the game, none of the agents has money in its bank account. In order to procure parts, the agents have to borrow money from the bank. The agents have to pay interest over the borrowed money. Money enters the bank account when customers pay for the shipped PCs. Money is deducted from the bank account for the procured parts and penalties for late deliveries to customers [4].

Daily production and inventory management of the agents

In the TAC SCM game, all agents are equipped with an identical simplistic PC factory that is managed by the game-server. The factory has one assembly cell that is capable to produce any type of PC. The factory has also a warehouse that stores the components and the finished PCs. The factory can only produce PCs for which the required number of components are available. PCs in the production schedule are processed sequentially until the capacity has been exhausted. At the end of each game day, the finished PCs are moved to the inventory, ready to be shipped to the customer the next day.

In the warehouse are stored PCs and components that are not shipped to the customers or used in production the next day. Holding costs have to be paid over the inventories. The agents have to minimize the inventories to reduce costs. An agent that is not able to fulfill customer demand will miss potential revenue. This makes the TAC SCM game a realistic supply chain management game. The game scenario is challenging but compact enough to model in a agent-based computer model [4].

The above section explained the game. The next section evaluates the software architecture of the MinneTAC agent.

3.2 The software architecture of the MinneTAC agent

This section explains the software architecture of the MinneTAC agent. The MinneTAC agent is successfully participating in the competition since 2003[34]. The next chapter explains the price prediction mechanism of the MinneTAC agent.

The software of the MinneTAC agent is developed in such a way that the interactions with the game-server and the behavioral components are clearly separated. This enables individual researcher to work separately. The MinneTAC agent has minimal dependencies among the individual components. The basis of the MinneTAC agent are the agent-ware software packages, distributed by the TAC SCM game organization, and the Apache Excalibur component framework. The Apache Excalibur framework provides the tools and standards to build the MinneTAC agent [34].

The Apache Excalibur is a general purpose component framework. Servers and middle-ware use this software as a foundation. In the MinneTAC agent, the Apache Excalibur framework provides the basis for a set of role-based configurable components. The components have well defined dependencies on the framework itself. A component is composed of a number of classes. Each Excalibur component is designed to fulfill a specific role. Every role is assigned to a specific Java class in the MinneTAC agent. A role has a set of responsibilities, a name and a well defined interface. When the agent starts the a TAC SCM game the configuration files load the specific roles and the Java classes and parameter settings.

The MinneTAC agent is a set of components layered on the Excalibur container. The Procurement, Sales, Production and Shipping components are responsible for the major decisions in the game. The Oracle component hosts a number of smaller components that are responsible future price prediction, market reports and inventory models. The Repository component is responsible for the communication between the components. The architecture of the MinneTAC agent completely separates the major decision making mechanisms. Evaluators are used to communicate between the components of the MinneTAC agent. The Evaluators are accessible through the Repository. The components of the MinneTAC agent have to make decisions, therefore the available data is inspected and an utility maximization has to determine the optimal decisions. The data is stored in the Repository. Repository data is encapsulated in the form of Evaluations. These Evaluations are used by the components to perform analysis and make decisions. The components are explained below [34].

The Communication component

The Communication component is responsible for the communication with the game-server. This component implements the software classes from the TAC SCM agent-ware packages that are responsible for the communication with the game server. The Communication component sends the incoming messages from the game-server to the Repository component. The Repository is responsible for the internal communication in the MinneTAC agent [34].

The Repository component

The Repository component is responsible for storing all the data that has to be shared among the components in the MinneTAC agent. Therefore, the Repository is visible to all other components in the agent. At the beginning of each game day, incoming messages from the game-server are placed in the Repository. The Procurement, Sales, Production and Shipping component retrieve the message from the Repository. The components perform an analysis, make decisions and communicate the decisions back to the Repository. The Repository sends the messages containing all the decisions made by the components to the Communications component [34].

The Oracle component

The Oracle component is a framework to set up a set of small configurable components. These components are used to perform analysis and future price prediction tasks [34].

The Sales component

The Sales component is responsible for selling the PCs to the customers. The agent has a limited factory capacity. Therefore, the agent has to bid on requests for quotes that maximize the expected profit. The allocation Evaluator in the MinneTAC agent calculates sales quotas for every PC type that maximizes the expected profit. A second price evaluator calculates, given the sales quotes, an order probability function that determines the prices the MinneTAC agent has to offer to the request for quotes from the customers to sell the PCS [34].

The Procurement component

The Procurement component is responsible for the procurement of new parts. Target inventory levels have to be maintained to meet customer demand. The procurement component has to anticipate future customer demand in order to have enough components available for production. The Procurement component sends RFQs to the suppliers and has to decide which offers to accept [34].

The Shipping component

The Shipping component is responsible for the delivery of finished PCs to the customers. PCs have to arrive on time to avoid penalties. Each game day, the Shipping component decides which finished PCs have to be shipped to the customers [34].

The production component

The Production component manages the factory. The production component has to maintain target inventory levels in order to meet customer demand[34].

3.3 Price prediction in TAC SCM agents

This section gives an overview of the price prediction mechanism of competing TAC SCM agents. The MinneTAC research team is testing techniques that can be implemented in real world decision support systems. Therefore, the MinneTAC agent is equipped with a flexible price prediction mechanism based on economic regimes [35]. Since the beginning of the TAC SCM game, the MinneTAC agent is consistently reaching the finals. On the other side, there are TAC SCM agents with more tailored price prediction mechanisms researching the finals. This section shows how competing agents are predicting the future market prices and how the MinneTAC agent differentiates itself from its main competitors.

The Botticelli TAC SCM agent

The Botticelli TAC SCM agent, developed by the Brown university, uses stochastic models to predict the future prices of the 16 different computer types it sells [36]. In other words, the sample average approximation method for stochastic discrete optimization is used. This is a Monte Carlo simulation based approach to stochastic discrete optimization problems. The basic idea of such methods is that a random sample is generated and also the corresponding sample averages function. When the obtained sample average function is solved, the procedure is repeated several times until a stopping condition is satisfied [36].

The Cmieux TAC SCM agent

The Cmieux TAC SCM agent is developed by the Carnegie Mellon university. The agent is build out of five modules. These modules are bidding, procurement, scheduling, strategy and forecasting [37]. The forecasting and bidding module are responsible for predicting the future demand and determining the offer price for customer orders. The first responsibility of the forecasting module is to predict the future customer demand. The strategy module uses this data to determine the target demand of the agent. The forecasting module uses the mean and trend values of the customer demand from past observations. The actual daily demand is drawn from a Poisson distribution. A separate Poisson distribution is used for each product. The forecasting module attempts to predict the changing mean and trend values from the different products governing demand separately. This is done with a linear least squares fit of observations form past game days [37]. The second responsibility of the forecasting module is to forecast the selling prices

of each product and the purchase price of the components of each product. This information is used by the other modules to make decisions based on the new market conditions. A linear least squares fit is calculated as well for the selling prices of each product over past game days. For every supplier, the purchasing prices are predicted with a nearest-neighbour technique based on historical prices. The forecasting module predicts the supplier prices for a particular day in the future by averaging the observed quotes with the nearby due dates[37]. The bidding module is for responding to the customer request for products. The goal of the bidding module is to sell the products for an as high as possible price. The bidding module has a probability distribution for each product that specifies the likelihood of winning the order. The distributions are learned off-line with data from previously played games to build a modified regression three called the distribution tree [37].

The RedAgent

RedAgent is developed by the School of Computer Science of the McGill University, Montreal, Canada. RedAgent is based on a multi-agent design, in which many simple heuristic-based agents manage tasks such as fulfilling customer orders or procuring particular resources [38]. The main idea is the usage of internal markets as the main decision making mechanism. Examples of decisions made are: what products are to be procured and how the existing resources are allocated. The internal agents communicate through a market mechanism in order to determine, collectively; which components are to be procured, which type of PCs are to be produced, how are the available components and production cycles allocated and what offers are to be send to the customers [38].

The DeepMaze agent

DeepMaze, a well performing TAC SCM agent, is developed by the university of Michigan. The price prediction mechanism consists of a nearest-neighbour machine learning algorithm. The algorithm identifies historical instances with similar economic indicators. This is augmented with an online learning procedure that transforms the predictors by an optimizing scoring rule [39]. This allows the selection of more relevant historical context using additional information available during individual games. For each day, the DeepMaze agent forecasts the price the customer market faces. On the current day, the request for orders from the customers is known and therefore the probability of winning an order can be calculated. To make predictions for future days, the agent combines the prediction of the distributions of the selling prices. The distributions are combined with the requests for orders that are generated from prices that the agent expects to receive for selling each additional PC on each possible day. The selling price distributions are based on a Bayesian network. The DeepMaze agent learns form historical data by clustering the indicators of certain market conditions with the likely distributions of current and future market prices with the k-nearest neighbours algorithm [39].

The TacTex agent

The TacTex agent is developed by the department of computer sciences of the university of Texas at Austin. TacTex is one of the best performing TAC SCM agents. The TacTex agent has two modules, that are responsible for performing all the tasks. The supply manager module is responsible for the procurement of the parts and whether to accept the offers from the suppliers. The demand manager module is responsible for bidding for customer orders, sending the daily production schedule to the factory and for the deliveries of the finished computers to the customers [40]. The Supply Manager handles all planning related to component inventories and purchases, and requires no information about computer production except for a projection of future component use, which is provided by the Demand Manager. The Demand Manager, in turn, handles all planning related to computer sales and production. The only information about components required by the Demand Manager is a projection of the current inventory and future component deliveries, along with an estimated replacement cost for each component used. This information is provided by the Supply Manager [40]. The TacTex agent uses machine learning techniques to train the price prediction mechanism with historical price data, collected from previous TAC SCM tournaments. The selected machine learning techniques was an additive regression with decision stumps, an iterative method in which a decision stump is repeatedly fit to the residual from the previous step [40]. Before the start of the game, the price prediction mechanism of the agent is bootstrapped. During the game, the agent uses an off-line learning mechanism what enables the agent to adapt to changing market conditions. The future customer demand is predicted with a Bayesian approach [40].

The Mertacor agent

The Mertacor agent is developed by the electrical and computer engineering of the Aristotle university, Thessaloniki, Greece. The agent uses the integration of techniques from the Operations Research literature, adaptive algorithms and statistical modelling [8]. The agent consists of four modules. The four modules are the inventory, procurement, factory and bidding module. Before the beginning of the game, the price prediction mechanism is trained with historical data. The training data for predicting the winning price of a bid came from the two 2005 TAC SCM semi-finals and the test data from the two 2005 TAC SCM finals. There were 38 attributes considered for predicting the future market prices and the probability of winning an order. Feature selection was performed to determine the best attributes to use for a predictor. The feature selection was done with multiple linear regression and backward elimination based on the f-statistic. Out of the Nine of the 38 initial predictors were used. Thereafter, neural networks, support vector machines and the m5 method were used to compare the predictive performance of the multiple linear regression model on the training set. The m5 method performed best and was therefore selected in the TAC SCM competition [8]. The future market prices were calculated on-line. The

agent uses two predictors. The first predictor is based on linear interpolation, with the assumption that the maximum paid price corresponds to the maximum customer offer reserve price. The second predictor is a scheme using the k-nearest neighbours algorithm. This method is used to calculate the probability of each bid place by the agent[8].

The PhantAgent agent

The PhantAgent is developed by the department of computer-science of the "Politehnica" university Bucharest, Romania. In the agent architecture, each task is assigned to a different module. The modules are the Computer Module, Component Module, and Factory Module. The Component Module handles component buying and estimation of component costs and does not need any information from the other modules. However, the component cost estimation strategy will greatly influence the way the Computer Module behaves. The Computer Module handles computer sales and depends on the Factory Module for knowing how crowded the factory is, on the Component Module for estimating the profit margin and on both modules for calculating how many computers can be offered. The Factory Module allocates the computer production and does not need information from the other modules, but its strategy must be compatible with the selling strategy implemented by the Computer Module[41]. PhantAgent employes a more practical selling strategy, which aims to keep the maximum factory utilization at all times. This strategy is executed as long the profit is positive. An major advantage of this strategy is that by bidding on a large number of customer offers, the agent becomes an important presence in the market, forcing other competitors to adopt a similar price policy in order to get orders[41]. PhantAgent calculates its offer price by starting from the best highest price of the past three days, and adjusting it with a factor that depends on the current factory utilization. In short, this factor will increase when the factory is too crowded since this is a sign of too many won bids and it will decrease when the factory is not utilized well enough [41].

Chapter 4

The ensemble price prediction mechanism of the MinneTAC agent

The price prediction mechanism of the MinneTAC agent is explained in this chapter. The MinneTAC agent uses observable microeconomic conditions to predict the future market prices. These observable microeconomic conditions are called economic regimes [42].

This chapter is divided in two parts. A general overview of the price prediction mechanism of the MinneTAC agent is given in the first part. The building blocks of the price prediction mechanism are explained in the second part.

4.1 An overview of the price prediction mechanism of the MinneTAC agent

Regression based price prediction approaches are widely applied in general. These approaches assume that the functional form of the relationship between the response and the explanatory variables has the same structure throughout the planning horizon [43]. The price prediction mechanism of the MinneTAC agent extends the functional form of the relationship between the response and explanatory variables throughout the planning horizon. The MinneTAC agent is able to detect changes in the relationship between the prices and sales volume over time and this enables the MinneTAC agent to forecast a broader range of market conditions. The improvements in the price predictions of the MinneTAC agent are worked out by the variability in market conditions and the non-functional relationship between prices and sales volume [44].

Economic regimes are the basis of the price prediction mechanism of the MinneTAC agent. The intuition behind economic regimes is that prices re-

flect future expectations in the demand and supply relationship [23]. This definition is extended with the property that economic regimes are distinguishable statistical patterns of historical sales data [42]. These statistical distinguishable patterns are used to identify the economic regimes [42]. The MinneTAC agent uses historical sales data from previous TAC SCM tournaments to identify the economic regimes [42].

There are three economic regimes in the economic literature [23]. These are scarcity, balance and oversupply. Scarcity indicates that there is more demand than supply for a commodity and balance means that demand equals supply. When oversupply occurs, there is more supply then demand for a product. The MinneTAC agent uses five economic regimes. These are extreme scarcity, scarcity, balance, oversupply and extreme oversupply. The economic regimes extreme scarcity and extreme oversupply were added to prevent outliers [42].

The MinneTAC agent uses the price prediction mechanism to make strategic and tactical decisions during the TAC SCM games. The MinneTAC agent determines the probability that customers will accept an offer. Furthermore, the MinnetAC agent has to allocate its resources in an as optimal as possible way. The economic regimes change during the game and this influences the tactical and strategic decisions of the MinneTAC agent. In the figure below, this is illustrated.

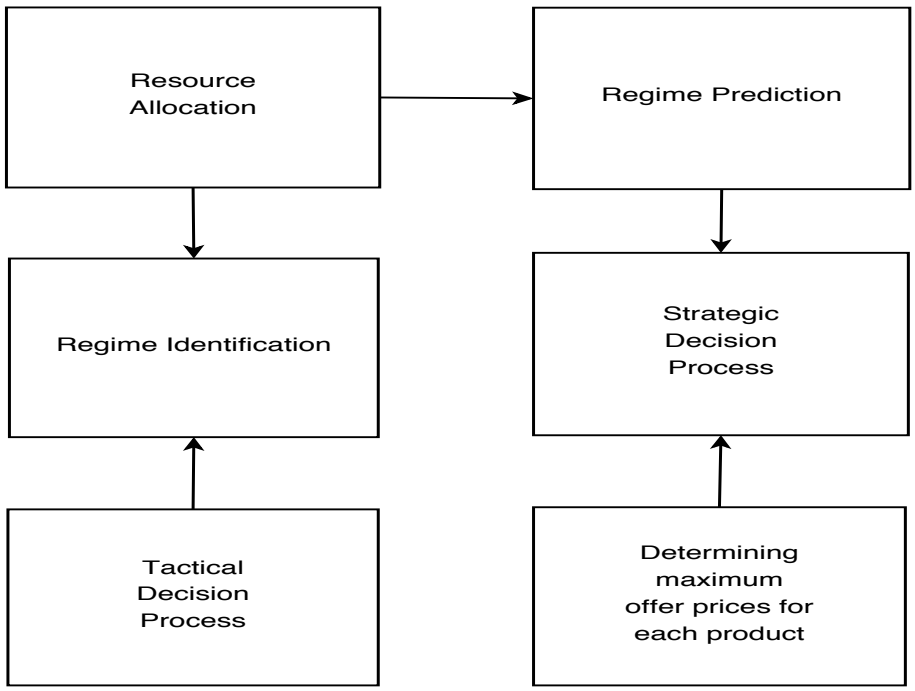

Figure 4.1: Strategic and tactical decisions during the TAC SCM game.

Strategic decisions include procurement of new components and schedul-

ing the available production capacity of the factory. Tactical decisions are concerned with determining the maximum prices the customers are willing to pay for the finished products. These tactical decisions have to be made within a certain time range determined by the strategic decisions [42].

An example is the procurement strategy. Procurement decisions are based on the expected future economic regimes. When customer demand is expected to grow in the future, new parts have to be procured to meet the future demand. When an increase in sales prices is expected, it is a thoughtless decision to sell aggressively with lower margins. A more intelligent approach is to keep the finished products in stock and wait for higher prices. This approach will increase the margin made on the products and the bottom line is that the profitability will increase.

An brief overview of the price prediction mechanism of the MinneTAC agent is given in figure 4.2., There is explained how the historical price data is processed to determine the economic regimes in the next section. A explanation is given how the MinneTAC agent predicts the future market prices during the game in the last section.

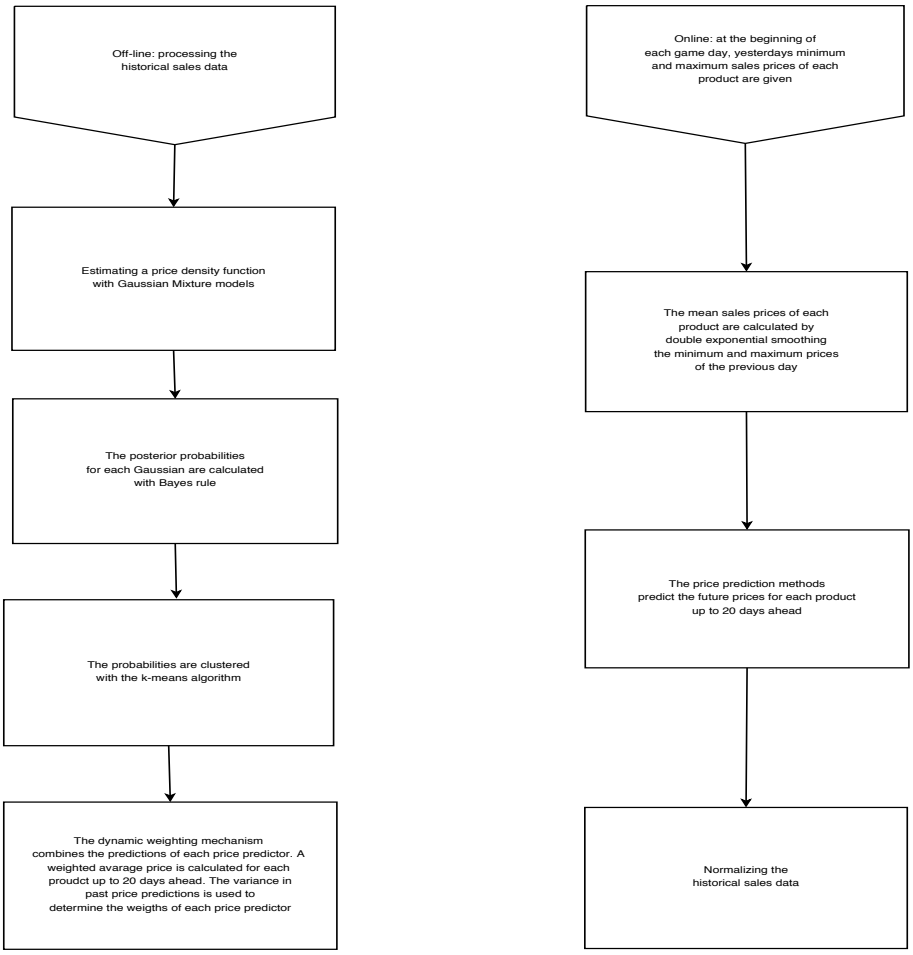

Figure 4.2: On overview of the price prediction mechanism.

4.2 The identification of the economic regimes

An explanation is given how the MinneTAC agent learns the economic regimes from historical sales data in this section. The historical sales is obtained from previously played TAC SCM tournaments.

The historical sales data is normalized in the first place. Normalization is applied because the sales data is obtained from 16 products that have different price ranges. Normalization also enables a comparison between different price ranges. The normalization is applied in (4.1),.

$$\mathrm{np} = \frac{\mathrm{price}}{\mathrm{assemblyCost} + \sum_{j=1}^{\mathrm{numParts}} \mathrm{nominalPartCost}_j} \tag{4.1}$$

np is the normalized price. The normalized price is based on the price of the product divided by the sum of the production and material costs.

After the normalization of the historical price data, a Gaussian Mixture

Model (GMM) is used to calculate the price density function and to determine the economic regimes. A Mixture Model is a useful method for density estimation. Gaussian Mixture Models are a very popular Mixture Model [45]. The Expectation Maximization (EM) algorithm is used to calculate the Gaussian components of the GMM. The EM algorithm is one of the most known statistical algorithms to find the mode of a marginal likelihood estimation or posterior distribution function [46]. A GMM is a semi-parametric estimation method since it defines a general class of functional forms for a density model. The goal of probability density estimation is to model a given set of training samples which approximates the true probability density function. Probability density estimation could be used for clustering, classification and novelty detection in addition to estimating a density function [47]. Other advantages of the semi-parametric approach of GMM models are fast computations and efficient memory usage [42].

The density of the normalized price is calculated in (4.2):

$$p(\mathrm{np}) = \sum_{i=1}^{N} p(\mathrm{np}|c_i)P(c_i). \tag{4.2}$$

$p(\mathrm{np}|c_i)$ is the i-th Gaussian from the GMM and $P(c_i)$ is the prior probability of the i-th Gaussian in this formula.

In (4.3), the $p(\mathrm{np}|c_i)$ is calculated:

$$p(\mathrm{np}|c_i) = p(\mathrm{np}|\mu_i, \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{\left[\frac{-(\mathrm{np}-\mu_i)^2}{2\mathrm{x}\sigma_i^2}\right]}. \tag{4.3}$$

μ_i is the mean and σ_i is the standard deviation of the i-th Gaussian from the GMM.

The choice of the number of Gaussians should reflect a balance between the predictive accuracy of the model and the computational overhead. Gaussian models are sensitive to over-fitting. Increasing the degree of freedom in the model leads to a higher change of over-fitting the data.

Bayes' rule is used to determine the posterior probability. This is performed in (4.4) [42]:

$$P(c_i|\mathrm{np}) = \frac{p(\mathrm{np}|c_i)P(c_i)}{\sum_{i=1}^{N} p(\mathrm{np}|c_i)P(c_i)} \qquad \forall i = 1, \ldots, N. \tag{4.4}$$

$P(\mathrm{np}|c_i)$ stands for the posterior probability of the i-th Gaussian. The posterior probabilities given the normalized price np are stored in a N-dimensional vector for all Gaussians. This is shown in (4.5) [42]:

$$\vec{\eta}(\mathrm{np}) = [P(c_1|\mathrm{np}), P(c_2|\mathrm{np}), \ldots, P(c_N|\mathrm{np})] \tag{4.5}$$

$\vec{\eta}(\text{np})$ is the N-dimensional vector of posterior probabilities for the $P(c_i|\text{np})$.

For each normalized price np_j, the vector of posterior normalized price probabilities, $\vec{\eta}(np_j)$, is calculated. $\vec{\eta}$ is evaluated for each normalized price np_j

The economic regimes are obtained by clustering the price distributions across days with the k-means algorithm with a similarity measure on the probability vector $\vec{\eta}(np_j)$ and the normalized prices np [42].

The clusters that are found represent the frequently occurring price distributions. The center of each cluster is a probability vector that correspond to a regime $r = R_k$ for $k = 1,..., M$, where M is the number of regimes. A conditional probability matrix, $P(c|r)$ is created by combining the probability vectors for each regime. This matrix has N rows for each component of the GMM and M columns for every regime [42].

The density of the normalized price np, dependent on the regime R_k is calculated in (4.6). $p(np|R_k)$ is the density of the normalized price np given a regime R_k, and $P(c_i|R_k)$ is the probability of the i-th Gaussian given the k-th regime [42]:

$$p(\text{np}|R_k) = \sum_{i=1}^{N} p(\text{np}|c_i)P(c_i|R_k). \tag{4.6}$$

The probability of regime R_k dependent on the normalized price np is computed with Bayes' rule in (4.7).

$$P(R_k|\text{np}) = \frac{p(\text{np}|R_k)P(R_k)}{\sum_{k=1}^{M} p(\text{np}|R_k)P(R_k)} \qquad \forall k = 1, \ldots, M \tag{4.7}$$

In (4.7), M represents the number of economic regimes and the prior probabilities, $P(R_k)$ of the different regimes are calculated by a counting process over past data.

A Markov transition matrix is computed to predict the future economic regimes during a TAC SCM game in the last step. This matrix represents the posterior probability of transitions at time step $t+1$ to regime r_{t+1} given the current regime r_t at time step t [42].

The economic regimes were identified with historical sales data. The MinneTAC agent has to identify the dominant economic regime in the TAC SCM games. By calculating the normalized prices for the current time step t in the game, the dominant economic regime is identified. The estimated normalized price at time step t is indicated by $\overline{\text{np}}_t$. The MinneTAC agent selects the economic regime with the highest probability. This is shown in (4.8):

$$\underset{1 \leq k \leq M}{\text{argmax}} P(R_k|\overline{\text{np}}_t). \tag{4.8}$$

4.3 How the game data is processed online

New information becomes available that affects the future market prices during TAC SCM games. The game server sends each participating agent a market report each game day. This is the most important information source of each agent. The market report gives the maximum and minimum sales prices for every product. This information is used by the price prediction mechanism of the MinneTAC to calculate the future prices of each product.

The price prediction mechanism of the MinneTAC agent requires the mean sales prices of each product to forecast the future prices. The problem is that the market report has no information about the quantities sold of each product. This makes it very difficult to accurately calculate the average prices of each product. A weighted average price based on the minimum and maximum prices has serious drawbacks [42]. The lack of accuracy of this method is a problem. This is caused by fluctuations in minimum and maximum prices of each product. Outliers could occur that are not within the true distribution of the prevailing prices. Therefore this method is not used by the MinneTAC agent [42]. Double exponentially smoothing the minimum and maximum sales prices of each product solves the lack of accuracy of taking a weighted average price [42].The double exponential smoothing is performed in 4.9) and (4.10).

A Brown linear i.e. double exponential smoothing method smoothes minimum and maximum prices to lower the impact of sudden price changes. Equations (4.9) through (4.11) show how to smooth $\mathrm{np}_{d-1}^{\min}$ using double exponential smoothing with an α of 0.5. The result of exponentially smoothing is referred as $\widetilde{\mathrm{np}}_{d-1}^{\min}$. The same procedure is applied to $\mathrm{np}_{d-1}^{\max}$ as well, which results in $\widetilde{\mathrm{np}}_{d-1}^{\max}$.

$$\widetilde{\mathrm{np}}_{d-1}^{\min'} = \alpha \cdot \mathrm{np}_{d-1}^{\min} + (1-\alpha) \cdot \widetilde{\mathrm{np}}_{d-2}^{\min'} \tag{4.9}$$

$$\widetilde{\mathrm{np}}_{d-1}^{\min''} = \alpha \cdot \widetilde{\mathrm{np}}_{d-1}^{\min'} + (1-\alpha) \cdot \widetilde{\mathrm{np}}_{d-2}^{\min''} \tag{4.10}$$

$$\widetilde{\mathrm{np}}_{d-1}^{\min} = 2 \cdot \widetilde{\mathrm{np}}_{d-1}^{\min'} - \widetilde{\mathrm{np}}_{d-1}^{\min''} \tag{4.11}$$

Because the quantities of the products are not given the average of the smoothed maximum and minimum price is taken. With the preceding day's normalized minimum price $\widetilde{\mathrm{np}}_{d-1}^{\min}$ and normalized maximum price $\widetilde{\mathrm{np}}_{d-1}^{\max}$:

$$\widetilde{\mathrm{np}}_{d-1} = \frac{\widetilde{\mathrm{np}}_{d-1}^{\min} + \widetilde{\mathrm{np}}_{d-1}^{\max}}{2}. \tag{4.12}$$

In this formula:

- α is the smoothing constant and has a value of 0.5
- $\widetilde{\mathrm{np}}_{d-1}^{\min}$ is the minimum price of the previous day

- $\widetilde{\text{np}}_{d-1}^{\max}$ is the maximum price of the previous day
- $\widetilde{\text{np}}_{d-2}^{\min}$ is the minimum price of the day before yesterday
- $\widetilde{\text{np}}_{d-2}^{\max}$ is the maximum price of the day before yesterday

It proved that this approach is more accurate than taking the smoothed average of the minimum and maximum prices [44].

4.4 The ensemble price prediction mechanism of the MinneTAC agent

The concept of ensemble price prediction was developed to improve price predictions. Ensemble price prediction combines multiple price prediction methods with different properties. The combination of multiple price prediction methods has to compensate the shortcomings of each individual price prediction method. A dynamic weighting mechanism has to determine the weights of each price prediction method in such a way that the shortcomings of each prediction method are compensated. The weights of each price prediction method are based on the inverse of the variance between previously predicted prices in the MinneTAC agent. The variance measures the accuracy of the price prediction mechanism of the MinneTAC agent [48].

The dynamic weighting mechanism is an important part of the ensemble price prediction mechanism of the MinneTAC agent. The dynamic weighting mechanism has to determine the weights for each price prediction method for multiple time horizons [49].

4.4.1 The price prediction methods

The MinneTAC agent uses a Markov, a Markov-correction, an Exponential smoother and Exponential-2C-adapter price prediction methods.

These price prediction methods are explained in the next sections.

The Markov price predictor

Stochastic processes evolve over time in a probabilistic manner. Markov chains are a special kind of stochastic processes. Markov chains have a special property that probabilities involving the future process only depends on the present state of the process and are independent of events in the past. In other words, this property says that the conditional probability of any future event given any past events and the present state is independent of the past events and depends only on the present state. The conditional probabilities for a Markov chain are called transition probabilities. When this probabilities are fixed, these are called stationary transition probabilities. Each time when a Markov chain is observed, the chain can be in any of the states.

Given the current state, a transition matrix gives the probabilities which state will be next [3]. In the case of the MinneTAC agent, the transition matrix is created by an off-line counting process [35]. This matrix contains the posterior probabilities of transition from day N, to day $N+1$ given the current economic regime on day N [35]. There are two different types of Markov prediction methods. The first is a Markov interval prediction (4.13). The interval is based on training a separate Markov transition matrix for each day in the planning horizon [42]. The second method is repeated one day prediction (4.14) and (4.15). This is done by repeating a single day prediction matrix. The MinneTAC agent is able to work with both methods.

Interval prediction:

$$\vec{P}(\vec{r}_{d+h}|\widetilde{\mathrm{np}}_{d-1}) = \sum_{r_d+h} \dots \sum_{r_d+h} \left\{ \vec{P}(\vec{r}_{d+h}|\widetilde{\mathrm{np}}_{d-1}) \cdot \mathrm{T}_n(r_{d+n}|r_{d+n-1}) \right\}. \quad (4.13)$$

Repeated one-day prediction:

$$\vec{P}(\vec{r}_{d+h}|\widetilde{\mathrm{np}}_{d-1}) = \sum_{r_d+h} \dots \sum_{r_d+h} \left\{ \vec{P}(\vec{r}_{d+h}|\widetilde{\mathrm{np}}_{d-1}) \cdot \mathrm{T}_n^{h+1}(r_d|r_{d-1}) \right\}. \quad (4.14)$$

where:

$$\mathrm{T}_n^{h+1}(r_d|r_{d-1}) = \prod_{n=0}^{h} \mathrm{T}_1(r_d|r_{d-1}). \quad (4.15)$$

The Markov-correction price prediction method

This price prediction method is used for long term, strategic, prediction of future economic regimes. The prediction part is the same as the above mentioned Markov predictor, but the entire price history is taken into account. So instead of the normalized yesterdays price $\widetilde{\mathrm{np}}_{d-1}$ we use $\vec{P}(\vec{r}_{d-1}|\widetilde{\mathrm{np}_1}, \dots, \widetilde{\mathrm{np}}_{d-1}\})$, to indicate a vector of the posterior probabilities of all the regimes on day $d-1$.

This method is based on two distinct operations. The first operation is a recursive Bayesian update of the posterior probabilities for regimes based on the historical measurements of smoothed mid-range normalized price since the time of measurement until the previous day. The second distinct operation is a prediction of the posterior distribution of regimes N days in the future. This is done recursively as in the Markov predictor [44].

The Exponential smoother price predictor

The exponential smoother predictor is an alternative for the Markov based price prediction methods [42]. In the first step, the price trend, $\widetilde{tr}_{d-1}$, is calculated.

$$\widetilde{tr}_{d-1} = \frac{a}{1-a} \cdot (\widetilde{\mathrm{np}}'_{d-1} - \widetilde{\mathrm{np}}''_{d-1}) \quad (4.16)$$

in this formula:

- a = the exponential smoothing constant.
- $\widetilde{\text{np}}'_{d-1}$ = the single smoothed mid-range normalized price series obtained by applying simple exponential smoothing to the normalized mid-range prices.
- $\widetilde{\text{np}}''_{d-1}$ = the double-smoothed normalized price series obtained by applying simple exponential smoothing to $\widetilde{\text{np}}'_{d-1}$.

After calculating the smoothed mid-range price using the daily minimum and maximum price, this step is also applied for the price trend.

$$\widetilde{\text{tr}}_{d-1} = \frac{\widetilde{\text{tr}}^{\min}_{d-1} + \widetilde{\text{tr}}^{\max}_{d-1}}{2} \tag{4.17}$$

- $\widetilde{\text{tr}}^{\min}_{d-1}$ is the exponentially smoothed minimum normalized trend.
- $\widetilde{\text{tr}}^{\max}_{d-1}$ is the exponentially smoothed maximum normalized trend from the previous day.

The maximum smoothed price is calculated in the same way. The mid-range price range is calculated in the same way as the mean price with (4.12).

Using the calculated mid-range trend and yesterdays price estimate we estimate today's and the future daily smoothed prices as:

$$\widetilde{\text{np}}_{d+n} = \widetilde{\text{np}}_{d-1} + (1+n) \cdot \widetilde{\text{tr}}_{d-1} \qquad \forall n = 1, \ldots, h. \tag{4.18}$$

The density of the normalized price dependent on the regime R_k is obtained with the formula below:

$$p(\widetilde{\text{np}}_{d+n}|\hat{R}_k) = \sum_{i+1}^{N} p(\widetilde{\text{np}}_{d+n}|\zeta_i) P(\zeta_i|R_k). \tag{4.19}$$

Then we use the formula:

$$P(\hat{R}_k|\widetilde{\text{np}}_{d+n}) = \frac{p(\widetilde{\text{np}}_{d+n}|\hat{R}_k) P(R_k)}{\sum_{k=1}^{M} p(\widetilde{\text{np}}_{d+n}|\hat{R}_k) P(R_k)} \qquad \forall k = 1, \ldots, M. \tag{4.20}$$

to compute the predicted probability dependent on the predicted exponentially smoothed normalized price n days into the future np_{d+n}..

The Exponential-2C-adapter

The exponential-2C-adapter is a price prediction method, based on exponential smoothing. The exponential-2C-adapter is independent of economic regimes. The main function of this price prediction method is to predict prices for one day ahead. The main reason for implementing the exponential-2c adapter price predictor is the accuracy for one day ahead price predictions.

4.4.2 The dynamic weighting mechanism

The dynamic weighting mechanism of the MinneTAC agent was developed by Wolfgang Ketter et al. [50]. The dynamic weighting mechanism has to combine the predictions of each price predictor. The weighting mechanism is called dynamic because the weights of each price predictor are constantly updated during the game. The weights are based on the inverse of the variance in past price predictions. The variance indicates how accurately each price prediction method predicts the future prices. A low variance means accurate price predictions. The MinneTAC agent predicts the future market prices for one up to 20 days ahead. This procedure is performed for each product at each game day. The weighting mechanism calculates the optimal weights of each price prediction method up to 20 days ahead. Price prediction methods that performs optimal for short term predictions obtain a higher weight for short term predictions compared to predictors that are more suitable for further ahead predictions [42].

The price prediction methods have an equal weight in the ensemble when TAC SCM games starts. The dynamic weighting mechanism starts learning the optimal weights of each price predictor after the second game day.

An explanation is given how the dynamic weighting mechanism calculates the weight of each price prediction method in the sections below.

The MinneTAC agent is configured with a block weighting and an exponential smoothing variance calculation method. These methods are explained in the sections below.

The block weighting variance calculation method

The block weighting variance calculation method is comparable with a moving average of the most recent N number of price predictions. The number of past price predictions is called the block weight horizon and is configurable in the MinneTAC agent. The configuration file is shown in the appendix. The size of the number of previously calculates variances determines the influence of new observed variances on the weights of each price prediction method. A small block weight horizon is more reactive to new observed variances than a large block weight horizon [33].

$$\sigma_{i,n}^2 = \frac{1}{N} \sum_{j=T-N+1}^{T} \left(\widehat{\mathrm{np}}_{i,n}(j) - \widetilde{\mathrm{np}}_{i,n}(j)\right)^2 \tag{4.21}$$

In this formula:

- T = current day.
- N = Block length, i.e. number of past predictions.
- $\widehat{\mathrm{np}}_{i,n}$ = predicted value of np forecasted by predictor i.

- $\widetilde{\text{np}}_{i,n}$ = real time estimated value of np.

The exponential smoothing variance calculation method

Exponential smoothing is a widely applied method to predict stationary time-series [33]. The current variance is the weighted average of the new calculated variance and the past variances. The smoothing constant τ determines the relative weight placed on the current variance. The τ value is also configurable in the configuration file of the MinneTAC agent. A small τ value means that the variance is more reactive to changes in the predictive performance of each price prediction method. Hence, exponential smoothing applies a set of declining weighs to all previously calculated variances. In (4.22) and (4.23), the variance is calculated [42].

$$\sigma_{i,n}^2 = C \cdot \sum_{j=1}^{T} \exp(-\frac{T-j}{\tau})(\widehat{\text{np}}_{i,n}(j) - \widetilde{\text{np}}_{i,n}(j))^2 \tag{4.22}$$

$$C = \frac{1}{\sum_{j=1}^{T} \exp(-\frac{T-j}{\tau})} \tag{4.23}$$

In this formula:

- T = current day.
- τ = Smoothing constant.
- $\widehat{\text{np}}_{i,n}$ = predicted value of np forecasted by predictor i.
- $\widetilde{\text{np}}_{i,n}$ = real time estimated value of np.

The weighting mechanism

After calculating the variance of each price prediction method, the optimal weights are determined. In (4.24) and (4.25), the optimal weights of each price predictor are calculated, where $\widehat{\text{np}}_n$ is the normalized market price on day n over a planning horizon h. $\widehat{\text{np}}_1, n$ is the predicted normalized price for predictor 1 on day n, and $\omega_{1,n}$ presents the normalized weight for predictor 1 on day n.

$$\widehat{\text{np}}_n = \omega_{1,n} \cdot \widehat{\text{np}}_{1,n} + \omega_{2,n} \cdot \widehat{\text{np}}_{2,n} + \ldots + \omega_{S,n} \cdot \widehat{\text{np}}_{S,n}, \qquad \forall_n = 1, \ldots, h \tag{4.24}$$

The confidence of individual predictions, i, on day ,n, with the variance error resulting from formula (4.21), is determined by the following formula:

$$\psi_{i,n} = \frac{1}{\sigma_{i,n}^2} \tag{4.25}$$

Summing the individual weights $\psi_{i,n}$, and dividing them by the sum of the weights Ω_n, resulting from formula (4.26):

$$\Omega_n = \frac{1}{\sigma^2_{1,n}} + \frac{1}{\sigma^2_{2,n}} + \ldots + \frac{1}{\sigma^2_{S,n}} \tag{4.26}$$

results in the normalized weights:

$$\omega_{1,n} = \frac{\frac{1}{\sigma^2_{1,n}}}{\Omega_n} \tag{4.27}$$

4.5 Summary

The main purpose of this chapter was to provide an insight in the the price prediction mechanism of the MinneTAC agent. The next chapter explains the experimental design and methodology to improve the price prediction mechanism of the MinneTAC agent. The knowledge of this chapter enables a better understanding of research explained in chapter 5 and 7.

Chapter 5

The improvements in the ensemble price prediction mechanism of the MinneTAC agent

The research related to improving ensemble price prediction in the MinneTAC agent is explained in this chapter. The concept of ensemble price prediction was developed to improve price predictions [48]. Ensemble price prediction tries to make more accurate price predictions by compensating the disadvantages of individual price prediction methods. A weighting mechanism has to determine the weights of each price prediction method in such a way that the shortcomings are compensated. In the MinneTAC agent, the weights of each price prediction method are based on the inverse of the variance between previously predicted prices. The variance measures how well the prices were predicted. The variance can be calculated with a block weighting or an exponential smoothing method [48].

Experiments and empirical research are used to answer the research questions. The outline of the experimental design is as follows. There is started with an explanation of the experiments that are performed to benchmark the predictive performance of the MinneTAC. Thereafter, the methodology to answer the research questions is explained.

5.1 The experimental design

The experiments are performed in a special computer environment, developed by the university of Minnesota [51]. This computer environment simulates TAC SCM games. The MinneTAC agent is competing with the best performing TAC SCM agents in these experiments. A special feature of this computer environment is the option to control the market conditions during TAC SCM games. This enables a more profound comparison of the price

prediction mechanism of the MinneTAC agent. The game server environment is therefore called a controlled game server. During these games, the performance of the MinneTAC agent is measured. After each change in the price prediction mechanism, new games are performed to benchmark the performance of the MinneTAC agent.

The experiment manager is a software tool to manage and orchestrate the experiments. The experiment manager is located on the game servers of the university of Minnesota[51]. In figure 5.1, an overview of the TAC SCM game environment is given.

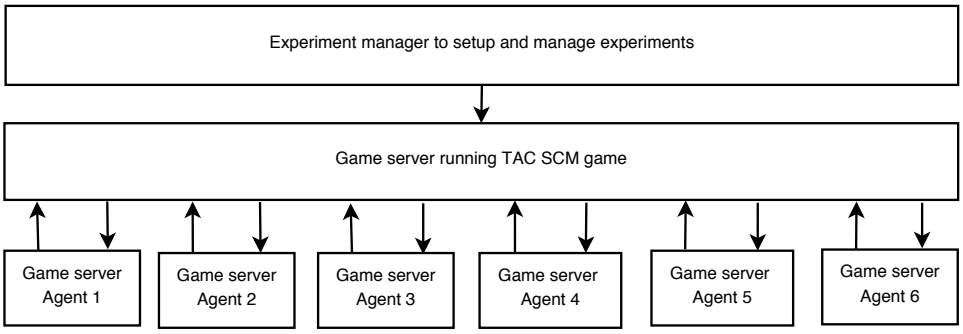

Figure 5.1: An overview of the TAC SCM game environment, developed by the university of Minnesota.

The game environment consists of seven servers. Each participating agent requires a separate server and one server is used to run the TAC SCM game. The working of the TAC SCM game was explained in chapter 3.

The experiments are configured in the experiment manager. To setup an experiment, the following settings have to be made:

- The name of the experiment.
- The number of TAC SCM games the agents have to play.
- The configuration of the MinneTAC agent that is used.
- Which game servers to use for the experiments.
- The competitors of the MinneTAC agent
- The game seeds to play the games or to randomly select the game seeds.

When the TAC SCM games are finished, log files are stored on the servers of the university of Minnesota. Valuable information about the game and the participating agents is stored in these log files. The MinneTAC also makes log files in which specific information about the performance of the MinneTAC agent is stored. The information in the log files of the MinneTAC

agent is used to analyze the predictive performance of the price prediction mechanism.

Multiple games are performed to measure the performance of the MinneTAC agent. Section 5.1.1 explains the required number of experiments to come to statistically significant results.

Multiple configurations of the MinneTAC are available. The preferred configuration of the MinneTAC agent has to be selected before each experiment. Currently, there are multiple configurations of the MinneTAC agent available.

A feature of the experiment manager is to manually assign servers to each participating agent. Some agents have a more complex software architecture and need a more powerful server to run optimally. The MinneTAC agent has a relative complex price prediction mechanism and needs a powerful server to function optimally.

In advance, the competitors of the MinneTAC agent have to be selected. The MinneTAC agent is competing with the best performing agents in the recent TAC SCM competitions in the experiments. The same competitors are chosen to minimize the profile variability between the experiments. In the experiments, the MinneTAC agent is competing with the TAC TEX agent developed by the university of Texas in 2007, DeepMaze2007 developed by the university of Michigan, Mercator2008 developed by the Aristotle University of Thessaloniki, Crocodile2005 developed by the University of Zagreb and PhantAgent2007 developed by the University of Bucharest.

Game seeds are repeatable pseudo-random sequences of any individual market factor or combinations of market factors that influence the market variability in TAC SCM games [51]. In other words, game seeds are market situations that occur in TAC SCM games and the sequence in which they occur. A special feature of the TAC SCM game environment is to specify the game seeds in advance of an experiment. The game seeds have to be specified for each TAC SCM game. In the experiments with the MinneTAC agent, for each game, the same game seed is used to overcome the randomness between the experiments. The experiment manage could also executed experiments with random game seeds [51].

5.1.1 The required number of experiments

The simulations of dynamic markets like the TAC SCM game have a high complexity and a high variability. The performance of the MinneTAC agent is dependent on the market conditions and the strategies of its competitors. Because the space for possible strategies and market conditions is very large, it is difficult to evaluate the performance of the MinneTAC agent. The TAC SCM game environment of the university of Minnesota, with predefined game seeds, enables a comparison between different experiments. With 40

games per experiment, an accurate and statistically significant comparison is made [51].

Figure 5.2 shows the differences between the Regular game server for TAC SCM games, a demand controlled game server and the controlled game server of the university of Minnesota. The demand controlled game server is a game server that is also able to reduce the differences between different TAC SCM experiments [51]. The demand controlled game server is used to benchmark the game server of the university of Minnesota. 5.2 visualizes how many games per experiment have to be performed to come to a certain profit difference. The actual profit difference measures the difference between the games. As already described in chapter three, a TAC SCM game consists of 220 virtual days, each game day has a length of 15 seconds. Performing 40 games takes 55 hours. Because of this time constraint, in this research, for every change in parameter settings not more than 40 games are performed. This does not compromises the accuracy of the outcomes.

The game data is analyzed to measure the performance of the price prediction mechanism of the MinneTAC agent after the experiments are finished. The next sections explains how the game data is extracted from the log files.

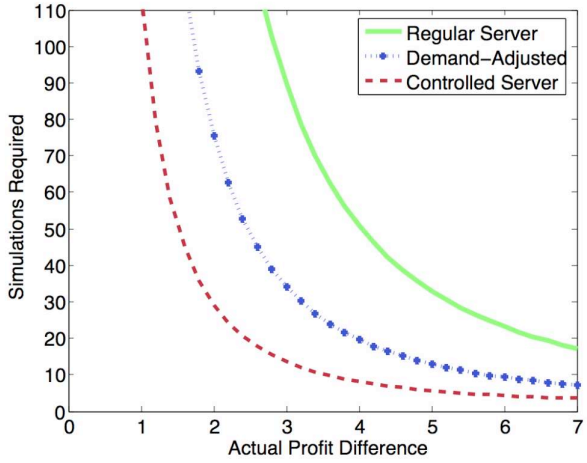

Figure 5.2: The difference between the regular, demand controlled and controlled game sever developed by the University of Minnesota [51].

5.1.2 How the game data is obtained and analyzed

Valuable game data is logged in TAC SCM games. The observed prices of each product during the games are an example of logged information. All the game data, generated by the MinneTAC agent and the game server are stored in zip files, located in the user directory on the Caesar server of the university of Minnesota. The game server saves game specific data including the real prices of each product on each virtual game day. These prices are

used to asses the predictive performance of the price prediction mechanism of the MinneTAC agent.

All valuable information specific to the performance of the MinneTAC agent is logged by the MinneTAC agent. This logged data contain information about the actions and performance of the MinneTAC agent. The MinneTAC agent writes all this information to a log file. This log file is also stored in a zip file in the user directory on the Caesar server of the university of Minnesota. The problem is how to obtain the required values out of the log file. In the first place, the zip file has to be unzipped. Because there are 40 games per experiment, a shell script automatically extracts the log files. Thereafter the required data needs to be extracted from the log file. This is done with the help of regular expressions, implemented in Python scripts. Again, a shell script is used to automate this extraction process. The extracted game data is converted to comma separate values that can be read easily by statistical or mathematical computer programs to analyze the data.

Matlab is a matrix based mathematical computer program to analyze the game data [52]. For each experiment, all the price data is read into Matlab. The price data is converted into matrices in Matlab. With these matrices, the mathematical analysis is performed. This process is implemented in Matlab scripts that are able to automatically perform these above mentioned steps. The outcomes of the analysis are in turn saved in a new matrix. With this matrix, it is possible to make tables and plot graphs to visualize the outcomes from the experiments.

To extract the required game data from the log file of the game server, a similar approach is used. The required data is obtained after these steps. This data is ready to process in a statistical or mathematical computer program.

The next section explains how the performance of the price prediction mechanism is measured.

5.1.3 The performance measures

The performance of the MinneTAC agent is measured between the predicted future prices of the price prediction mechanism and the prices occurred during the TAC SCM games. The root mean square error (RMSE) is used. The RMSE is also called the root mean square deviation (RMSD) [53]. The RMSE is a frequently used measure to determine the differences between predicted values of a model and the values actually observed from real data [53]. The RMSE is calculated between the predicted normalized price, $\widehat{\mathrm{np}}_n$ and the actual normalized price, np_n, over a prediction interval n between the current day and the end of the planning horizon, h, to determine the accuracy of the price prediction. The values of the RMSE have to be as

small as possible[51].

$$RMSE(\widehat{\mathrm{np}}_n, \mathrm{np}_n) = \sqrt{\frac{\sum_{\gamma=1}^{N_G} \sum_{d=1}^{N_D-n} (\widehat{\mathrm{np}}_d^{n,\gamma} - \mathrm{np}_d^{n,\gamma})^2}{N_G \cdot (N_D - n)}}, \qquad \forall n = 1, \ldots, h \quad (5.1)$$

- N_D = the length of the TAC SCM game in days.
- N_G = the number of test games.
- $\widehat{\mathrm{np}}_n$ = the predicted normalized price.
- np_n = the actual normalized price.
- n = prediction interval.

5.2 The methodology to improve ensemble price prediction

The methodology to improve the price prediction mechanism of the MinneTAC agent is explained in this section. The methodology is divided in four parts. The experiments with the aim to find optimal quantity of historical price predictions to calculate the variance as accurately as possible are explained in the first part. The experimental design was explained in the previous section. The variance could be calculated with two methods based on moving averages or exponential smoothing. The optimal weights of each price prediction method in the ensemble are calculated after finding the optimal τ value of the exponential variance calculation method, and the optimal number of historical price predictions of the block weighting variance calculation method. With the optimal weights of each price predictor, the MinneTAC agent can be bootstrapped. Bootstrapping is explained in the third part. Bootstrapping is a form of supervised learning [11]. Bootstrapping tries to avoid the costly learning process of the optimal weights of each price predictor in the beginning of the TAC SCM games. During recent TAC SCM competitions, the MinneTAC agent's performance increased during the games after the optimal weights were learned. The ensemble prices predicted by the MinneTAC agent are benchmarked with the prices predicted by the individual price prediction methods in the last part of the methodology. Ensemble price prediction tries to predict more accurate prices by compensating the disadvantages of the individual price predictors. Are the disadvantages of each price predictor correctly compensated in the MinneTAC agent?

5.2.1 The optimal quantity of historical price predictions

The weights of each price prediction method are based on the variance in past price predictions. How many past price predictions are needed to calculate to calculate the variance as accurately as possible? Experiments with

different quantities of historical price data are executed to answer these research questions.

When to much historical price predictions are used, the price prediction mechanism is over-fitted. The problem with using too little historical price data is that price predictors are not well enough trained and therefore lack accuracy. Over-fitting means that a statistical model describes the random error or noise in the data instead of the underlying relationship. Over-fitting is important when a learning algorithm is trained with samples for which the desired output is known. The algorithm may adjust to specific random features in the data that have no causal relation to the training data [45].

The variance can be calculated with a block weighting method based on moving averages or an exponential smoothing method. An explanation is given how the experiments with these methods are performed below.

The exponential smoothing method

In section 4.4.2, formulas (4.22) and (4.23) explained the exponential variance calculation method. The exponential smoothing parameter τ determines to optimal quantity of historical price predictions to calculate the variance. The τ determines the relative weight placed on the current observed variance compared to the previously calculated variances. In other words, with a small τ, the variance is more reactive the changes in the accuracy of each price prediction method. This implies that the weights of each price predictor change more rapidly. What is the optimal reactivity of the variance to calculate the future prices as accurate as possible?

The block weighting method

In section 4.4.2, formula 4.21 explains the block weight variance calculation method. The block length N is varied in the block weighting formula. The block length is the number of historical price predictions that are taken into account to calculate the variance of each price predictor.

Which value of the block weight horizon leads to the most accurate future price predictions? The block weight horizon is varied between an interval of 5 to 30 days of historical price predictions in the experimentes. With a low value for the block weight horizon, the weights of each price prediction method are more reactive to changes in the predictive accuracy. With a high value for the block weight horizon, the weights of each price predictor are less reactive to changes in the predictive performance. How reactive has to price prediction mechanism to be to calculate the future prices as accurate as possible?

The MinneTAC agent has to possibility to configure which variance calculation method is used during the experiments. This is done in a xml-based

configuration file. The configuration file is included in the appendix. The experiments could be executed after selecting the variance calculation method and the optimal value of historical prices to calculate the variance. When the games are finished, the steps explains in section 5.1.2 are performed. This section explained how the game data is extracted from the log files of the game server and the MinnetAC agent.

The MinneTAC agent logs the price predictions of each price predictor and the ensemble price calculated by the weighting mechanism. The prices are predicted for and up to 20 days ahead for each virtual game day. The game server logs the real prices that occurred on each virtual game day. When the experiment is finished and the prices are extracted from the log files of the MinneTAC agent and the game server, the data is loaded in Matlab to determine the predictive accuracy of the MinneTAC agent. The Root Mean Square Error (RMSE) is used to determine the accuracy of the price prediction mechanism of the MinneTAC agent. The RMSE is calculated for each individual price predictor and the ensemble price determined by the weighting mechanism of the MinneTAC agent. The RMSE was explained in section 5.1.3.

The RMSE is calculated for each price predictor and the ensemble prices for one up to 20 days ahead in the future. The RMSE values are plotted in a graph to make a comparison between the accuracy of the price prediction methods. This graph has to show which price prediction method showed the best performance and if the ensemble price is more accurate than the price predicted by the individual prediction methods.

After the experiment is analyzed and the performance is determined, the τ and block weight horizon are changed and a new experiment is started. This procedure is repeated until the optimal values for the τ and block weight horizon are determined.

When the optimal values of the block weight horizon and τ parameter are found, the optimal weights can be calculated.

5.2.2 The optimal weights for each price prediction

The optimal weights of each price prediction method are calculated for three purposes:

1. The weights are calculated to determine if the ensemble consists of the optimal combination of price prediction methods. Are the disadvantages of each price prediction method compensated?
2. The optimal weights are used to bootstrap the MinneTAC agent.
3. The optimal weights are calculated to check if the price prediction mechanism of the MinneTAC agent is correctly working.

The weights of each price predictor are based on the inverse of the variance calculated with the exponential variance calculating method in formulas 4.22 and 4.23 or with the block weight variance calculation method in formula 4.21. In formulas 4.24, 4.25 and 4.26, the optimal weights are calculated with the variances from the exponential or the block weight method.

The weights of each price prediction method are calculated in Matlab by implementing the exponential and block weight variance calculating methods and formulas 4.24, 4.25 and 4.26. The prices predicted by the MinneTAC agent for 20 days ahead on each virtual game day and the real prices, extracted from the game server, are loaded in Matlab. The weights of each price prediction method are calculated for each game day. The weights are calculated for one up to 20 days ahead. These weights are stored and this procedure is repeated for each game day until the end of the game. Because there are 40 games performed at each experiment, these procedure is applied for each game. A weight matrix is built with the average weights of all experiments.

The MinneTAC agent predicts the future prices up to 20 days ahead. Therefore, the weight matrix will have 21 columns and four rows for each price predictor. The sum of the weights of each price predictor have to sum up to one in each column. In the TAC SCM competition, the agents trade 16 types of PCs. A separate weight matrix is calculated for each type of PC. This is done to obtain insight if there are difference between the different market segments of each PC type. The weight matrixes of each game are averaged in the last step. An example of an weight matrix is given in the table below.

Price predictor	day T	day $T+1$	day $T+N-1$	day $T+N$
1	0.6	0.4	0.3	0.2
2	0.2	0.4	0.3	0.3
3	0.2	0.2	0.4	0.5
sum of the weights	1	1	1	1

The weight matrices are plotted to obtain insight in the weight movements of each price predictor during the games. The weight matrixes containing the averages weights of each price predictor for all the 16 products. Thereafter, plots are made of the weights of each price predictor for the individual products.

A comparison is made between the weights calculated by the MinneTAC agent and the theoretically calculated weights in the last step. This step is performed to look if the weighting mechanism of the MinneTAC agent is correctly working. The weights assigned to each price predictor are logged in the MinneTAC agent for each game day. The weights are extracted from the logs files with regular expression implemented in Python scripts. This procedure is the same as obtaining the price data from the logs files of the MinneTAC agent. Plots are made to visualize the differences between the

weights calculated in Matlab and the weights calculated by the MinneTAC agent.

Bootstrapping the MinneTAC agent

Bootstrapping is a form of supervised learning [11]. The goal of bootstrapping is to avoid a costly learning process at the beginning of TAC SCM games. The performance of the MinneTAC agent increased during recent TAC SCM competitions after the optimal weights were learned.

Research in private communication with dr. Wolf Ketter, has identified that the first phase of the game ends after approximately 30 virtual game days. Start-up effects occur during the first 30 days of TAC SCM games. Start-up effects are characterized by insufficient supplies of computer parts from the suppliers. The participating agents are generally not able to meet the demand of the customers. This results in higher market prices for each type of PC traded. The start-up effects disappear after approximately 30 game days. The MinneTAC agent is bootstrapped with the weights for the first 30 days of the game to incorporate the start-up effects.

The weights of each price prediction method are calculated on exactly the same way as of weights for the whole length of the game. The difference is that only the first 30 days of each game are used instead of the whole game of the game to calculate the weights.

The price prediction mechanism of the MinneTAC is extended to incorporate bootstrapping. The current version of the MinneTAC agent's Java code is extended. A matrix is programmed with the optimal weights of each price predictor in the Java code of The MinneTAC agent. The optimal weights are loaded and assigned to each price prediction method when the MinneTAC agent starts.

A linear transition mechanism was build in the MinneTAC agent to enable a smooth transition between the bootstrap weights and the weights calculated by the weighting mechanism. This mechanism slowly starts to mix the bootstrap weights with the weights calculated by the weighting mechanism. The transition phase takes 10 virtual game days. Thereafter, the weights are exclusively calculated by the weighting mechanism.

The performance of the MinneTAC agent, with bootstrapping, is compared with the performance of the MinneTAC agent without bootstrapping. Experiments are performed with bootstrap weights calculated with the exponential variance and the block weighting variance calculation method.

The RMSE is used to measure the performance of the MinneTAC agent [43]. The RMSE values of bootstrapping the MinneTAC agent are compared with the RMSE values without bootstrapping. A comparison between bootstrapping with exponentially calculated weights and weights calculated with

block weighting variance calculation is also performed.

This chapter explained the experimental design and methodology to improve ensemble price prediction in the MinnTAC agent. The next chapter explains the results from improving ensemble price prediction in the MinneTAC agent.

Chapter 6

The outcomes of improving ensemble price prediction.

The outcomes from the experiments to improve ensemble price prediction in the MinneTAC agent are discussed in this chapter. The outline of this chapter is as follows. The optimal quantities of historical price predictions for the exponential and block weighting variance calculation methods are explained in the first section. The optimal weights of each price prediction method in the ensemble are evaluated in the next part. The results from bootstrapping the MinneTAC agent with the optimal weights for the first phase of the TAC SCM game are explained in the last part.

6.1 The optimal variances calculations

The predictive performance of the MinneTAC agent is measured with the variance between the predicted and the real prices during the experiments. The dynamic weighting mechanism uses the variance to determine the weights of each price predictor in the ensemble. An exponential and a block weighting variance calculation method were used by the MinneTAC agent. Section 4.4.2 explained the exponential and block weighting variance calculation methods. The outline of this section is as follows. A discussion of the outcomes from experiments with the exponential variance calculation method is below. An evaluation of the block weighting variance calculation method is given at the end of this section.

6.1.1 The exponential variance calculation method

The τ parameter determined the emphasis on the current observed variance compared to previously calculated variances. Experiments with τ values between 1 and 10 were executed. The weights of each price predictor did not change with τ values higher then 10. This implied that the τ parameter places to less emphasis on the current observed variance to change the weights of each price predictor in the ensemble. 6.1 shows the RMSE values

of the ensemble price calculated by the dynamic weighting mechanism of the MinneTAC agent. The RMSE values were calculated for each experiments with τ values varying between 1 and 10.

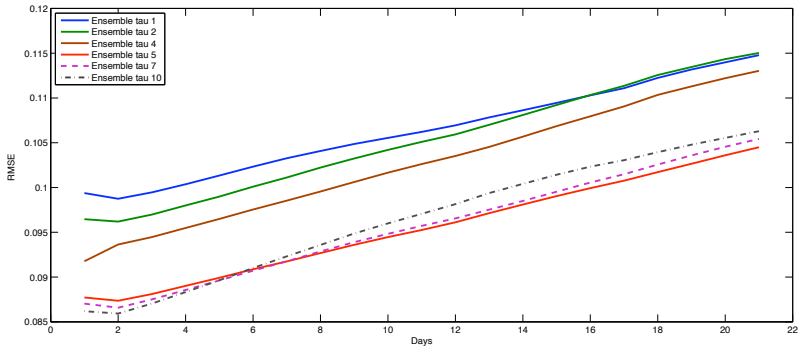

Figure 6.1: The RMSE values of the ensemble price with different τ values.

The y-axis shows the RMSE values of each experiment. The smaller the RMSE value, the more accurate the future prices were predicted. The RMSE was determined for price predictions from one day up to 20 days ahead. 40 TAC SCM games were performed with each τ value to come to statistically significant answers . The average RMSE value was used to determine the optimal τ. The x-axis shows the number of days ahead the future prices were predicted. There can be concluded that the experiments with a τ value of 5 showed the best performance. An overview of the standard deviations between the different τ values can be found in the appendix. The differences between the different τ values are marginal. The optimal τ values have a smoothing factor in the range between 4 and 10. Although the differences between τ values 1 and 5 are significant. An ANOVA analysis between the RMSE values of the experiments with τ 1 and 5, to test if the differences between the RMSE values were significant, are in the table below.

Source	**SS**	**df**	**MS**	**F**	**Prob F**
Columns	0.00126	1	0.00126	44.73	0.000005
Error	0.00112	40	0.00003		
Total	0.00238	41			

The RMSE values of each price predictor, for all experiments, are shown in the next graphs. An insight is provided in the differences between the predictive performance of the individual price prediction methods for all τ values.

The RMSE values of the experiments with all τ values for the Exponential price prediction method are shown in 6.2.The main findings is that there is not a clear τ value performing significantly better. This implies that the emphasis on historical price data did not influenced the predictive

performance of the exponential price prediction method.

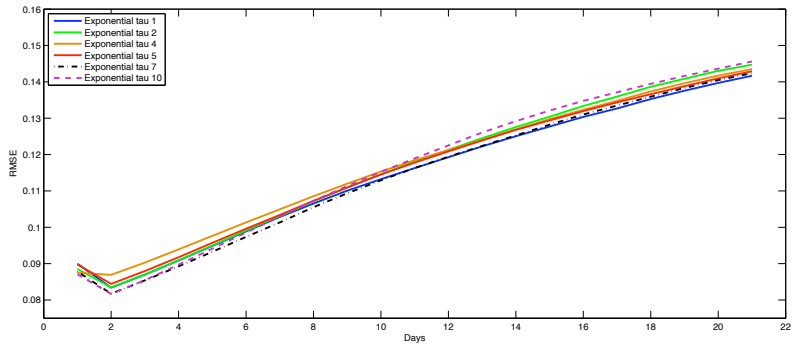

Figure 6.2: The RMSE values of the exponential price prediction method with exponential variance calculation.

The RMSE values of the exponential-2c-adapter are shown in 6.3. The differences between the different τ values are modest but there can be concluded that a τ of 5 showed the best performance.

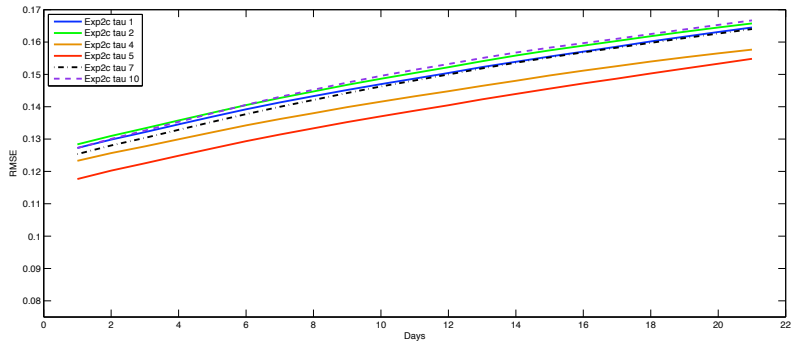

Figure 6.3: The RMSE values of The exponential-2c price prediction method with exponential variance calculation

The RMSE values from the experiments with the Markov price prediction method are shown in 6.4. The differences between the experiments are modest. The experiments with τ values of 1 and 7 showed the best performance. The relationship between these experiments is not clear and there can be concluded that the optimal τ values are in the range between 1 and 10. Furthermore, the decline in the predictive performance of the Markov predictor compared to exponential and exponential-2c-price predictors is smaller. This indicates that the Markov predictor performed more accurately for further ahead price predictions.

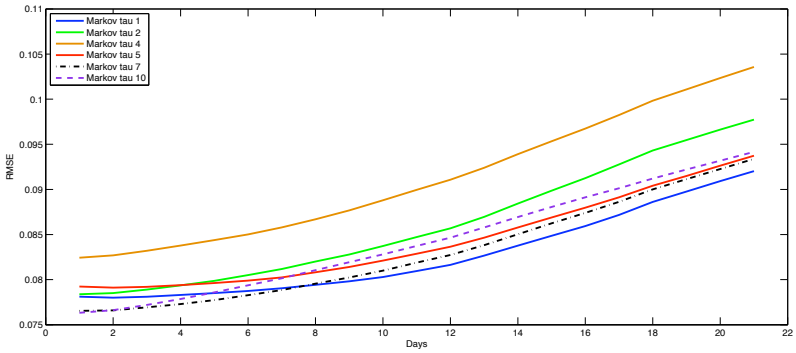

Figure 6.4: The RMSE values of the Markov price prediction method with exponential variance calculation.

The RMSE values of the Markov-correction price predictor are shown in 6.5. The outcomes are comparable with the outcomes of the Markov predictor. The main finding is that the optimal τ value is in the range between 1 and 10 days of historical price predictions.

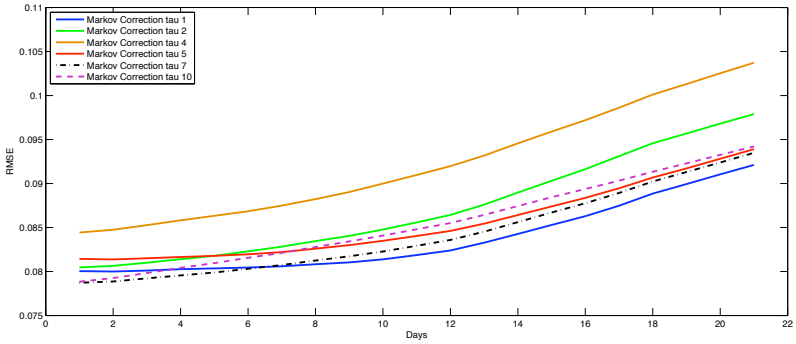

Figure 6.5: The RMSE values of the Markov-correction price prediction method with exponential variance calculation.

The conclusion of the experiments with exponential variance calculation is that the optimal τ values are in the interval between 1 and 10 and that on average a τ value of 5 showed the best performance. Although the differences are not statistically significant. Furthermore, a τ value of 5 lies in the middle of the range of τ values. The RMSE values of each price predictor with a τ value of 5 are shown in 6.6. The ensemble price, calculated by the dynamic weighting mechanism, is compared with the individual price prediction methods. The idea of ensemble price prediction is that the ensemble price has to compensate the disadvantages of each individual price prediction method. This has to lead to more accurate price predictions.

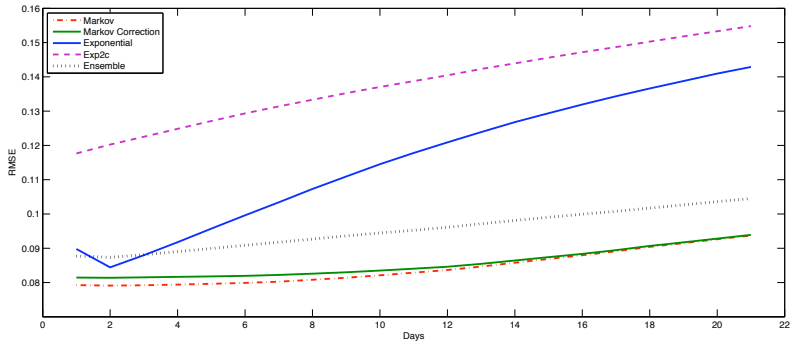

Figure 6.6: The RMSE values of each price predictor and the results of benchmarking the ensemble price.

The average RMSE values of 40 TAC SCM games, performed with τ 5, are shown on the y-axis. The x-axis shows the number of days ahead the MinneTAC agent predicted the prices. There main finding is that the Markov and Markov-correction price prediction methods showed the best performance for price predictions for one up to 20 days ahead. Furthermore, the price predictions of the Markov and Markov-correction predictors were more accurate then the ensemble price calculated by the dynamic weighting mechanism. This outcome is contrary to the expected outcomes and theory of ensemble price prediction.

In the next section, the theoretical weights of each price predictor are compared with the weights calculated by the dynamic weighting mechanism of the MinneTAC agent. A comparison between the theoretical and real weights has to provide insight why the ensemble price is less accurate than the prices predictions of the Markov and Markov-correction predictors.

6.1.2 The block weighting variance calculation method

The block weight horizon was varied between 5 and 30 days of historical price predictions in the experiments. The block weighting variance calculation method was explained in 5.2.1. The block weight horizon is the number of historical price predictions to calculate the variance. The weights of each price prediction method are more reactive to changes in the predictive performance with a low block weight horizon. Experiments with a block weight horizon higher then 30 days of historical price data were not executed. Experiments, with a block weight horizon higher than 30, showed no significant changes in the weights of each price predictor. The results of the experiments with block weight horizon values between 10 and 30 days of historical price data are below. The RMSE values of the ensemble price are in 6.7.

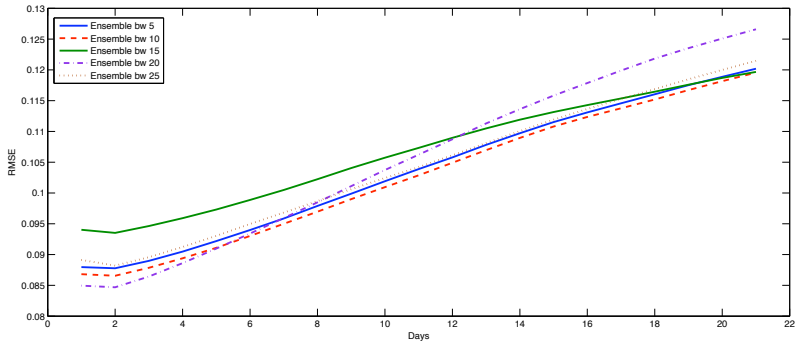

Figure 6.7: The RMSE values of the experiments with the block weighting variance calculation method.

The y-axis shows the average RMSE values of each experiment. 40 games per experiment were executed to come to statistically significant answers. The x-axis displays the number of days ahead the future prices were predicted.

Although the differences between the experiments are marginal and not statistically significant, the outcomes from the experiments show that a block weight horizon of 10 resulted in the most accurate future price predictions. An overview of the standard deviations between the different block weight horizon values can be found in the appendix. A significance test proved that there were no significant differences between the RMSE values of the experiments with a block weight horizon of 10 and 15 days of historical price predictions. The performance of block weight horizons of 10 and 15 days of price data differed the most. No statistically significant differences were found between these experiments. An Anova analysis was performed to measure the differences between the RMSE values of block weight horizons 10 and 15. The results from the Anova analysis are in the table below.

Source	**SS**	**df**	**MS**	**F**	**Prob > F**
Columns	0.00017	1	0.00017	1.66	0.2049
Error	0.00402	40	0.0001		
Total	0.00419	41			

The predictive performance of each price predictor was separately analyzed for each value of the block weight horizon. Insight was obtained in the differences between the individual price prediction methods. The exponential price predictor with different values of the block weight horizon is shown in 6.8. The values of the block weight horizon did not had a significant impact on the exponential smoother price predictor. There were only marginal differences between the experiments. There can be concluded that any value in the range of 5 to 30 days of historical data can be chosen for the exponential price predictor.

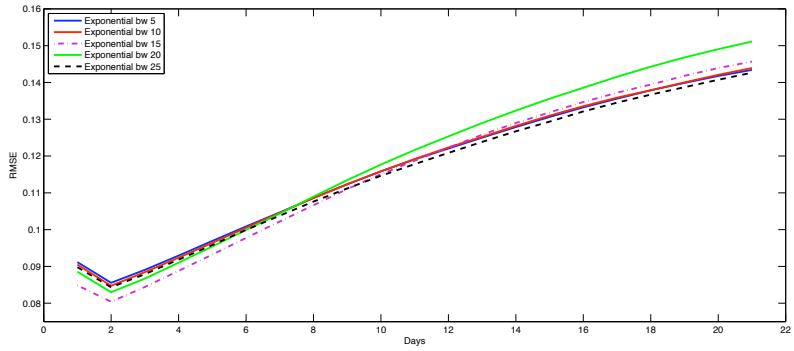

Figure 6.8: The RMSE values of the exponential price prediction method for all experiments the Block weighting variance calculation method.

The RMSE values of the exponential-2c-price prediction method are shown in 6.9. Although the differences between the experiments are marginal, a block weight horizon of 10 showed the best performance.

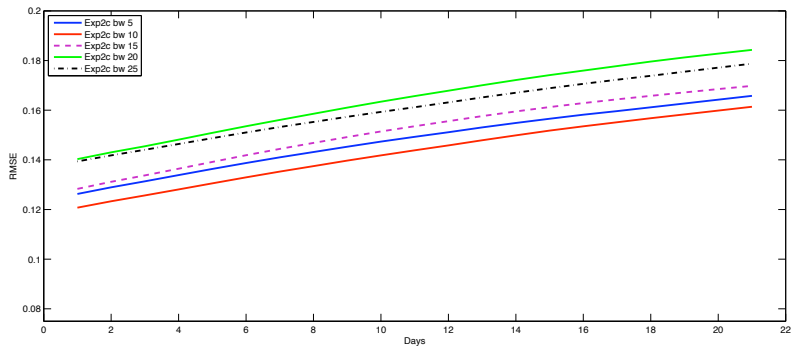

Figure 6.9: The RMSE values of the exponential-2c-price prediction method with the Block weighting variance calculation.

The RMSE values of the Markov price prediction method are in 6.10. The differences between the experiments are very marginal and statistically not significant. A block weight horizon of 15 showed the best performance for short term price predictions. A block weight horizon of 10 showed the best performance for price predictions from 8 up to 20 days ahead. The main finding is that the differences between the values of the block weight horizon are marginal. This implies that using a block weight horizons between 5 and 30 days of historical price predictions could not lead to large deviations in the predictive performance of the MinneTAC agent. A block weight horizon of 10 days of historical price predictions showed the most consistent performance.

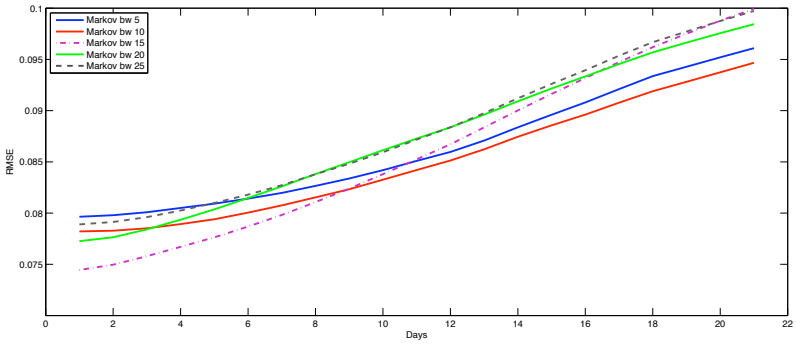

Figure 6.10: The RMSE of the Markov prediction method with the Block weighting variance calculation.

The experiments with the block weight horizon were initially performed with the exponential, exponential2c-adapter and Markov price prediction method. The Markov-correction price predictor was added to the ensemble. The Markov-correction price predictor had a very marginal influence on the predictive performance of the price prediction mechanism. 6.12 show the difference in the predictive performance of the ensemble with and without the Markov-correction price predictor. The predictive performance is measured with the RMSE and the experiments were performed with a block weight horizon of 10 days of historical price predictions.

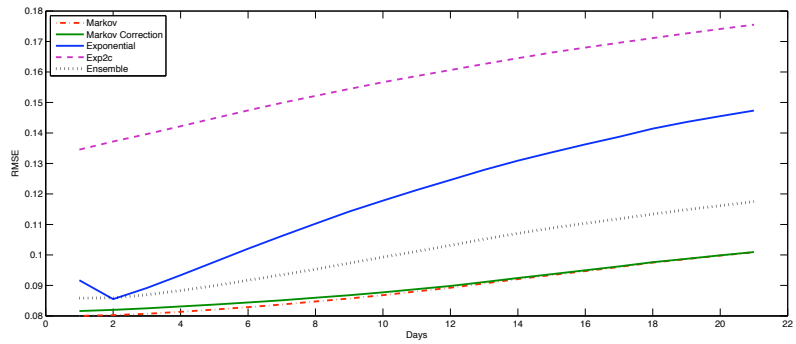

Figure 6.11: The results from benchmarking the ensemble price prediction system with block weighting variance calculation.

The ensemble price, calculated by the dynamic weighting mechanism, is benchmarked with the individual price prediction methods in 6.12. The Markov and Markov-correction price prediction methods showed a better predictive performance than the ensemble price. This is not conform the theory of ensemble price prediction. The ensemble price also did not outperform the individual price prediction methods with exponential variance calculation. After analyzing the theoretically calculated weights and the

weights calculated by the dynamic weighting mechanism, an analysis is made of possible causes of the performance of the ensemble price.

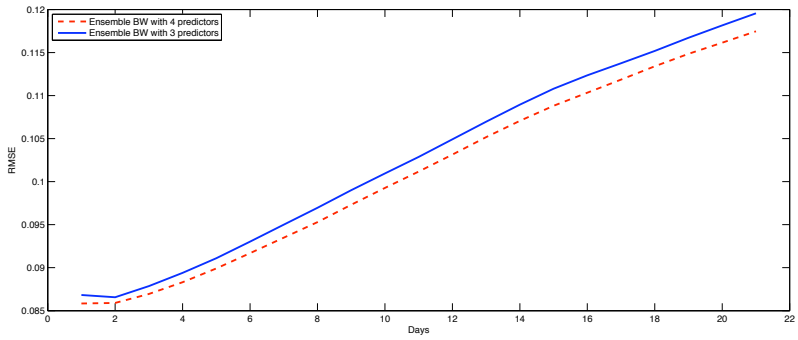

Figure 6.12: The results from benchmarking the ensemble price prediction system with block weighting variance calculation.

6.2 A discussion of the weights of each price predictor

A discussion of the theoretically calculated weights of the individual price predictors are given in this section. The theoretical weights of each price predictor were calculated with exponential and block weighting variance calculation. A block weight horizon of 10 day of historical price predictions and a τ of 5 were used. The formulas of the block weight and exponential variance calculation method were explained in 4.4.2. The prices, predicted by the MinneTAC agent, were compared with the real prices during these games. The theoretical weights were calculated to asses the mix of price prediction methods in the ensemble of the MinneTAC agent. Furthermore, the weights calculated by the MinneTAC agent were compared with the theoretical weights to assure that the weighting mechanism worked correctly.

6.13 shows the theoretical mean weights calculated with the exponential variance calculation method with a τ of 5. The number of days ahead the prices were predicted are shown on the x-axis. The mean weights of each price predictor, averaged over 40 TAC SCM games, are shown on y-axis. The y-axis shows the mean weights of each price predictor averaged over all 16 types of PCs in the TAC SCM game. The weights were averaged for each type of PC. The weights for each individual product are plotted in graphs in the appendix. The weights of each price predictor had only very marginal differences between the products. The weights of each price predictor are based on the inverse of the variance. The patterns in the weights movements of each price predictor are in line with the inverse of the RMSE calculated between the prices predicted by the MinneTAC agent, and the real prices during the TAC SCM games. The price predictors that had a better predictive performance had a higher weight in the ensemble. This

corresponds with the aim of the dynamic weighting mechanism. The less performing price predictors like the exponential-2c-adapter and the exponential price predictor still had a significant weight. This is contrary to the logic of ensemble price prediction.

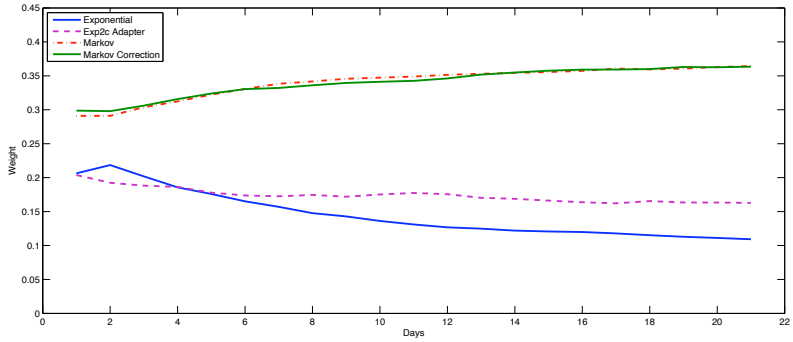

Figure 6.13: The mean weights with τ 5.

The theoretical weights, calculated with the exponential variance calculation method, are compared with the weights calculated by the MinneTAC agent in 6.14. The weights of each price predictor are logged for each game day for price predictions from one up to 20 days ahead in the MinneTAC agent. The theoretically calculated weights were compared with the weights of the MinneTAC agent. The theoretical weights and the average weights of the MinneTAC agent for game days 80, 100 and 160 are on the y-axis. The x-axis shows the number of days ahead the prices were predicted.

Large deviations between the weights of the MinneTAC agent and the theoretical weights were observed. The theoretical weights were more reactive to changes in the accuracy of each price predictor, with the same τ value. The weights of the MinneTAC agent changed only marginally from the standard weights assigned to each price predictor in the begin of each game. This means that there are differences between the weights calculated by the MinneTAC agent and the theoretical weights.

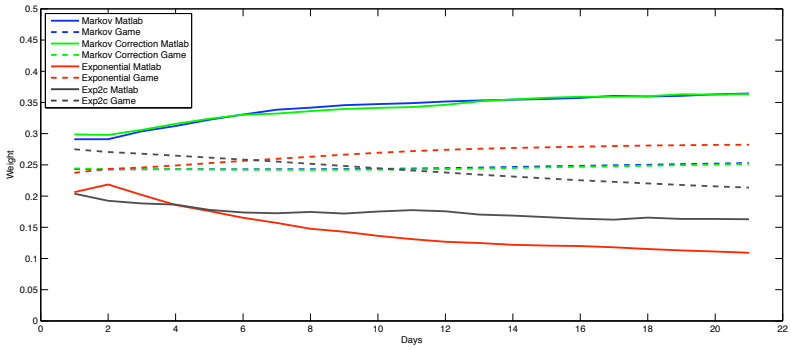

Figure 6.14: A comparison between the theoretical weights and the weights of the MinneTAC agent on the same days with τ 5

An explanation why the ensemble price did not outperform the Markov and Markov-correction price prediction methods could be that the and the exponential-2c-adapter had a significant weight. The exponential and exponential-2c-adapter predictors negatively influenced the ensemble price. The exponential and exponential-2c-adapter price prediction methods also had a significant weight in the theoretically calculated weights averaged over the whole length of the game. The ensemble price could also not outperform the prices predicted by the most accurate price prediction methods theoretically.

The theoretical weights calculated with the block weighting variance calculation method are shown in the next graphs. The mean theoretically calculated weights by the block weighting variance calculation method are in 6.15. The mean weights of each price predictor are on the y-axis. The mean weights of each price prediction method were separately calculated for each PC type over 40 TAC SCM games. A block weight horizon of 10 days of historical price predictions was used. The average weights of all 16 type of PCs traded in the TAC SCM games are on the y-axis. The weights for each individual PC type are plotted in the appendix. Only marginal differences between the weight movements were observed between the products. The number of days ahead the prices were predicted are shown in the x-axis.

The weights of each price predictor correspond with the predictive performance measured by the RMSE values. The weights of each price predictor reflect the predictive performance. This is corresponds with the concept of the dynamic weighting mechanism.

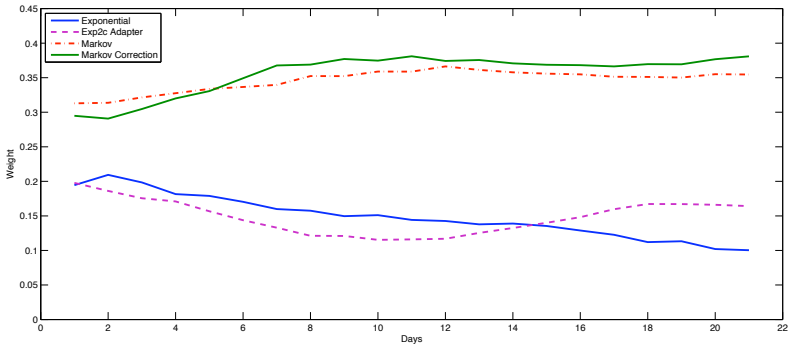

Figure 6.15: The theoretically calculated weights with a block weight horizon of 10.

The theoretical weights are compared with the weights calculated by the dynamic weighting mechanism of the MinneTAC agent in 6.16. The weights were compared on the same manner as with the exponentially calculated weights. The weights of each price prediction method are on the y-axis and the x-axis shows the number of days ahead the prices were predicted.

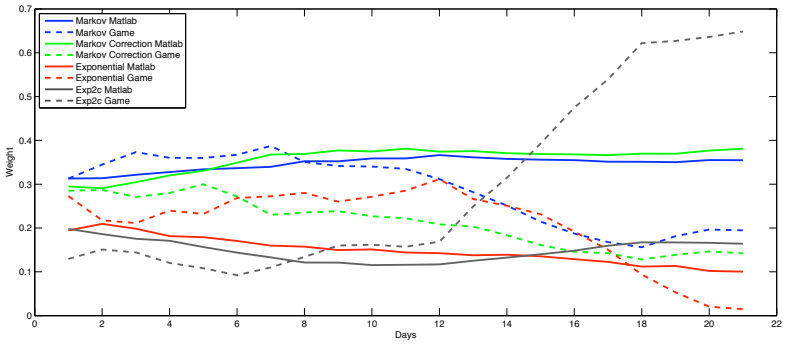

Figure 6.16: A comparison between the weights calculated by the MinneTAC agent and the theoretically calculated weights, with Blockweight 10

The weights calculated by the MinneTAC agent were more reactive than the theoretical weights. The weights, calculated by the MinneTAC agent with the block weighting method, also differed from the theoretical weights. The exponential-2c-adapter price predictor was assigned a higher weight than the Markov and Markov-correction predictor. The Markov and Markov-correction predictor outperformed the exponential-2c-adapter and this is contrary to the logic of the dynamic weighting mechanism. The predictive performance of the Markov and Markov-correction predictor is better than the exponential-2c-adapter for price prediction from 12 up to 20 days ahead. Future research has to investigate if the block weighting method was incorrectly implemented in the MinneTAC agent. The theoretical weights

show the same pattern as the weights calculated by the exponential variance calculation method. The exponential and exponential-2c-adapter price predictor had a significant weight while their performance lags behind the Markov and Markov-correction predictors in the theoretical weights. The exponential and exponential-2c adapter price predictors also had a significant weights in the exponentially calculated weights. The main finding is that the theoretical weights deviates from the weights calculated by the weighting mechanism of the MinneTAC agent. The weighting mechanism needs to be revised. A solutions is a weighting mechanism based on a voting mechanism. The most accurate price predictor is only used with a voting mechanism. A price predictor, that performs optimal for short term price predictions, is only used for short term price predictions. This weighting mechanism could use the exponential or block weighting variance calculation method to determine the predictive performance of each price prediction method in the ensemble.

An optimally composed ensemble has price prediction methods that perform optimal on different time frames and each prediction method has different characteristics. The exponential, Markov and Markov-correction prediction methods are based on economic regime price prediction. A solution is to add a price prediction method with more distinctive characteristics. Support Vector Machines or Neural Networks are able to handle large non-linear patterns [11]. The Markov-correction price predictor is an extension of the Markov predictor for further ahead price predictions. The Markov-correction predictor did not make more accurate price predictions for further ahead price predictions. This implies that the added value of the Markov-correction predictor is very marginal. Since the performance of the exponential-2c-adapter lags so far behind the other predictors, this predictor has to be replaced.

6.3 The discussion of the bootstrapped results

The outcomes from bootstrapping the MinneTAC agent with the optimal weights of each price predictor for the first game phase of TAC SCM games are discussed in this chapter.

The differences between the Minne TAC agent with and without bootstrapping with exponentially calculated weights are in 6.17. A modest increase in the predictive performance of the MinneTAC agent with bootstrapping is observed, although the increase in the predictive performance is not statistically significant. The average RMSE values of 40 TAC SCM games are on the y-axis. The number of days ahead the MinneTAC agent predicted the future prices is on the x-axis.

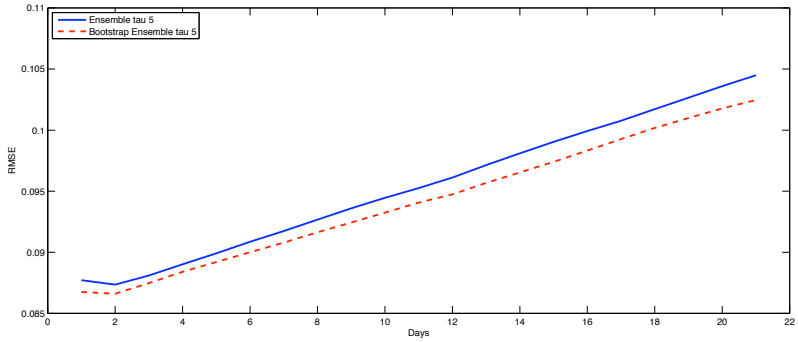

Figure 6.17: A comparison between the ensemble RMSE of bootstrapped and non-bootstrapped MinneTAC agent with exponential variance calculation.

The differences between bootstrapping and non-bootstrapping the MinneTAC agent over different game periods are in 6.18. The performance of the bootstrapped version of the MinneTAC agent was better at the beginning and end of the TAC SCM games. In the middle of the TAC SCM games, no major differences between the bootstrapped and non-bootstrapped versions of the MinneTAC agent were observed. The predictive performance of the MinneTAC agent slowly decreases during the TAC SCM games. Possible causes for the decrease in the predictive performance of the MinneTAC agent could be calculation of the weights of each price predictor and differences between the trained economic regime transition probabilities and the real economic regime transition probabilities in the games.

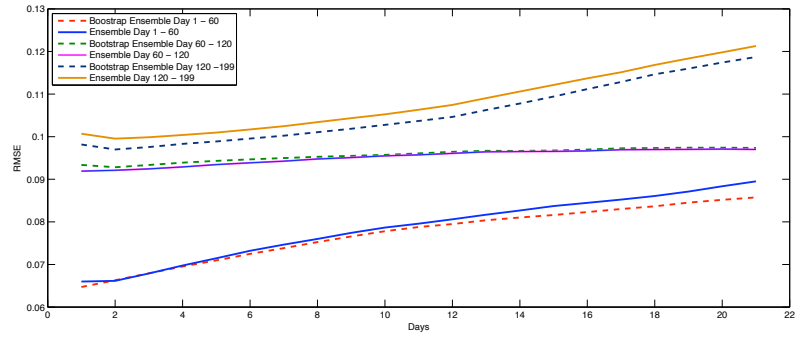

Figure 6.18: The differences between the MinneTAC agent with and without bootstrapping for different game phases. The exponential variance calculation method was used.

An ANOVA analysis was performed to test the statistical significance between the RMSE values of bootstrapping and non-bootstrapping the MinneTAC agent with the optimal weights calculated with the exponential variance calculation method. No statistically significant differences were found.

Source	**SS**	**df**	**MS**	**F**	**Prob > F**
Columns	0.00002	1	1.6389e-05	0.58	0.4515
Error	0.00113	40	.83448e-05		
Total	0.00115	41			

The results from bootstrapping the MinneTAC agent, with the weights calculated by the block weighting variance calculation method, for the first game phase of TAC SCM games are below.

The bootstrapped and non-bootstrapped version of the MinneTAC agent, with block weighting variance calculation, are compared in 6.19. The MinneTAC agent with bootstrapping showed a better predictive performance during the TAC SCM games although the differences are very small. Bootstrapping had a marginal effect on the predictive performance of the MinneTAC agent.

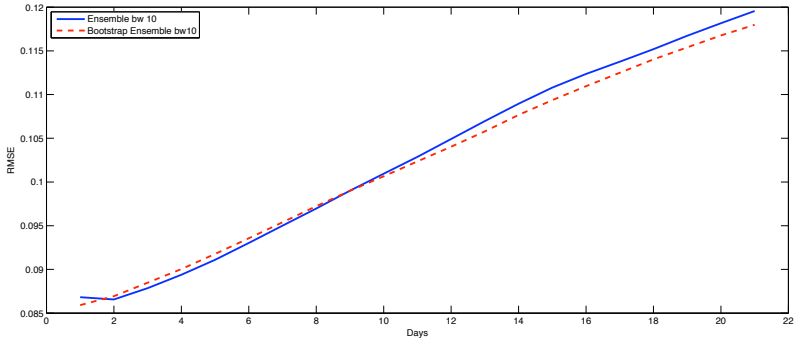

Figure 6.19: A comparison between the MinneTAC agent with and without bootstrapping with block weighting variance calculation. The performance was measured with the RMSE.

The predictive performance of bootstrapping and non-bootstrapping the MinneTAC agent, with block weighting variance calculation, for each games phase is in 6.20. The analysis was performed on the same way as with exponential variance calculation. The predictive performance also slowly decreased during the game with block weighting variance calculation. The outcomes were in line with the outcomes of exponential variance calculation.

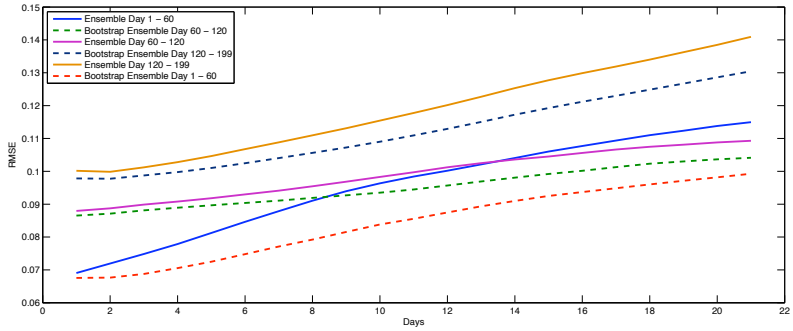

Figure 6.20:

The RMSE values of bootstrapping and non-bootstrapping with the optimal block weights did not resulted in statistically significant differences. An ANOVA analysis was used to indicate the statistical significance of the results in the table below.

Source	**SS**	**df**	**MS**	**F**	**Prob > F**
Columns	0.00001	0.00001	0.11	0.7449	
Error	0.00446	0.00011	.8		
Total	0.00447	41			

The performance of bootstrapping the MinneTAC agent with exponentially calculated weights and the weights calculated with the block weight variance calculation method is shown in 6.21.

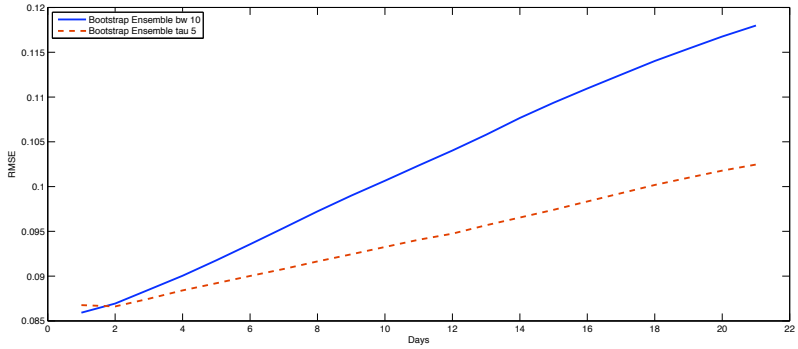

Figure 6.21: A Comparison between the RMSE values of bootstrapping with exponential and block weighting variance calculation.

The exponential variance calculation method outperformed the block weighting variance calculation method. This was in line with the differences in the predictive performance between exponential and block weighting variance calculation without bootstrapping.

Weight tables with the optimal weights of each price predictor with exponential and block weighting variance calculation are shown in the appendix in 8.3 till 8.10.

Chapter 7

Fuzzy economic regime transition probabilities in the MinneTAC agent

Fuzzy economic regime transition probabilities are implemented in this chapter. The MinneTAC agent uses economic regimes to predict the future market prices. An explanation how the economic regime transition probabilities were calculated was given in chapter four. The MinneTAC agent uses economic regime transition probabilities that are calculated over the whole length of a TAC SCM game. Previous research has shown that there are three distinctive game phases [54]. The price prediction mechanism of the MinneTAC agent is further improved with the optimal economic regime transition probabilities for each game phase. A Fuzzy smoothing mechanism has to enable smooth transitions between the different economic regime transition probabilities.

Fuzzy logic, developed by L.A. Zadeh in the mid-1960s, was developed for representing approximate knowledge that cannot be represented by conventional crisp Boolean logic [10]. Fuzzy logic is an extension of crisp bivalent logic since it is able to handle approximate knowledge. Fuzzy logic is based on fuzzy set theory. This enables an element to belong to a set at some degree and simultaneously not being part of the set at a complementary degree [10]. This allows a non-crisp (Fuzzy) relationship [10]. A fuzzy set is a set without clear or sharp boundaries or without a binary membership function. Partial membership of an element is possible in a Fuzzy set [10]. An example are old people. When could a person be defined as old? In a crisp set, an element belongs or does not belongs to the set. A Fuzzy set is represented by a membership function [10]. A particular element or value in the range of the fuzzy set has a grade of membership that indicates the degree of membership to the set [10].

This chapter is ordered as follows. An explanation about the different game phases is given in the first part. The fuzzy economic regime transition

mechanism is explained in the second part. The experimental design and methodology are explained in the third part. An evaluation of the results is in the last part.

7.1 The different game phases during the TAC SCM competition

Research has identified three game phases [54]. The economic regime transition probabilities may vary significantly during these game phases [54]. Start-up effects occur during the first 30 days of the game. The start-up effects are characterized by insufficient supplies of computer parts from the suppliers. The participating agents start with no inventories and in order to build PCs, the agent have to procure components. The suppliers also stat with no inventories. The participating agents are generally not able to meet the demand of the customers. This results in higher market prices for each type of PC. The start-up effects disappear after approximately 30 game days. The economic regime transition probabilities will change afterwards.

The market for PCs reaches a situation in which customer demand fluctuates randomly after 30 game days.

The end game effects start to occur after approximately game day 190. The participating agents start to reduce their inventories. The winner is the agent with the highest amount of money in its bank account. Inventories are not taken into account and have no value at the end of the game. The agents start to reduce their inventories by aggressively selling PC. This means that the market prices will be lower and the economic regime probabilities will inevitably change. The economic regime probabilities are recalculated to incorporate the excess of supply.

7.2 The Fuzzy economic regime transition mechanism

A fuzzy economic regime transition mechanism enables smooth transitions between the different economic regime probabilities. The economic regime transition process is described in figure 7.1. The economic regime transition probabilities change gradually over a period of 20 virtual game days.

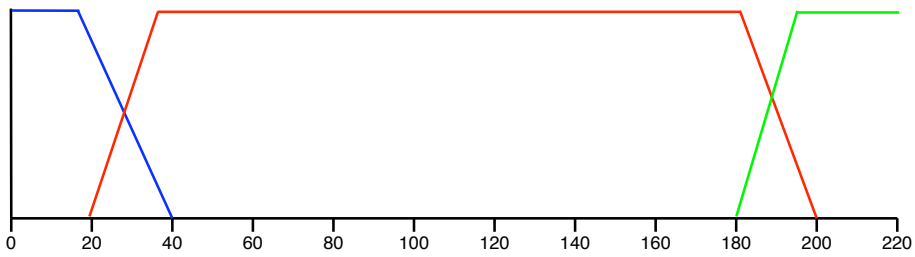

Figure 7.1: The gradual transition process between the different economic regime transition probabilities. The length of the TAC SCM game is displayed on the x-axis. The economic regime transition probabilities are shown on the y-axis. The blue line are the economic regime transition probabilities for the first game phase and the red and green line for the mid en end game phases.

A period of 20 days was used to change the economic regime probabilities. A gradual transition is required to enable the strategic and tactical decisions of the MinneTAC agent to adapt to the new transition probabilities. Strategic decisions include the procurement of new components and scheduling the available production capacity in the factory. Tactical decisions are concerned with determining the maximum prices the customers are willing to pay for the finished products. These decisions have to be made within a certain time range determined by the strategic decisions [42].

A linear and a sigmoid function are implemented in the MinneTAC agent to gradually change the economic regime transition probabilities. In advance, it is possible to select which is method is used during a TAC SCM game. These methods are explained below.

The sigmoid economic regime transition mechanism

A Sigmoid weighting functions is used to smooth the transitions between the different economic regime transition probabilities. The formula of the Sigmoid function is as follows:

$$f(x) = \frac{1}{1 + e^{-\alpha * x}} \tag{7.1}$$

In 7.2: $f(x)$ are the economic regime transition probabilities, and α is the slope parameter. An α greater than one makes the slope sharper and an α smaller than one makes the slope more smooth. An α of 1 was used in the experiments.

Figure 7.2 gives an overview of the transition process. On the x-axis, The length of the transition process is displayed between the different economic regime transition probabilities. The weights of each set op regime transition probabilities is displayed on the y-axis.

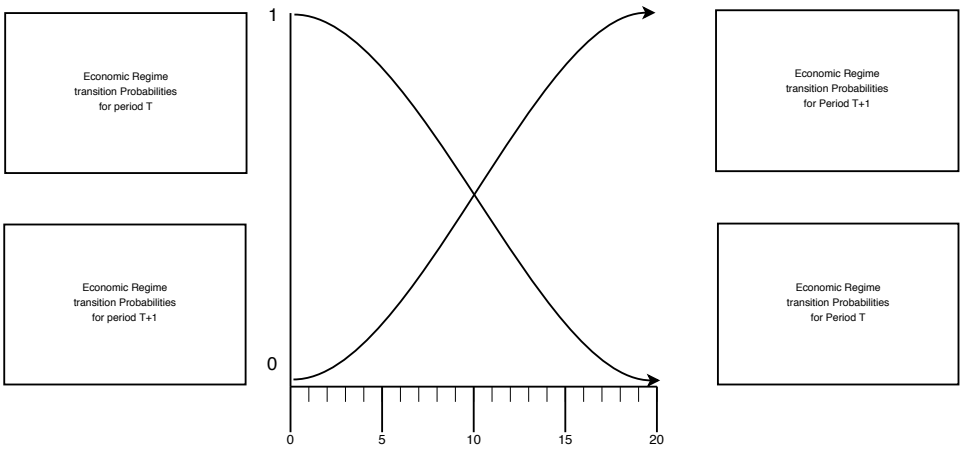

Figure 7.2: The Sigmoid function

The linear economic regime transition mechanism

The linear transition mechanism uses linear weights to provide a gradual transition between the economic regime transition probabilities. The transition process is explained in 7.3.

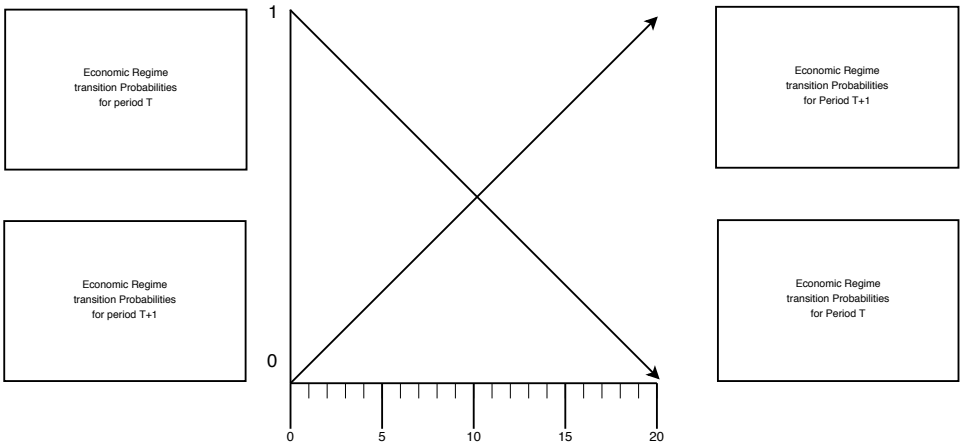

Figure 7.3: The linear economic regime transition mechanism.

On the x-axis, the length of the transition process is displayed between the different economic regime transition probabilities. The weights of each set op regime transition probabilities is displayed on the y-axis.

7.3 The experimental design and methodology

The economic regime mixer was implemented in 3 steps:

1. The economic regime transition probabilities were calculated for each game phase.

2. The Java code of the MinneTAC agent was extended with a Java class which mixes the economic regime transition probabilities.

3. The configuration file of the MinneTAC agent was adapted to select which fuzzy transition function the MinneTAC agent had to use during the TAC SCM games

The economic regimes of the MinneTAC agent are trained with historical sales data from previous TAC SCM tournaments. Section 4.2 explained how the economic regime transition probabilities were calculated. The training data was divided in three parts. The first part contained the historical sales data for the first 30 days, the second contained the sales data from day 30 until day and 190, and the third part the remaining sales data. The economic regimes were separately calculated with the formulas from section 4.2 to obtain the regimes for each game phase. This training data was loaded in a new version of the MinneTAC agent in order to execute the experiments.

The software of the MinneTAC agent was extended with a java class called RegimeTrendCombiner, and was implemented in the Oracle component of the MinneTAC agent. The sigmoid and the linear economic regime transition functions were implemented in this Java class.

The configuration file of the MinneTAC agent was adapted to configure which mixing function to use and to select the mixing period between the different game phases. The configuration file is in the appendix. Experiments were performed to asses to increase in the predictive accuracy of the price prediction mechanism of the MinneTAC with the customized economic regime transition probabilities. Experiments were executed with block weighting and exponential variance calculation.

The performance of the MinneTAC was measured with the Root Mean Square Error (RMSE) [43]. The RMSE was calculated between the prices predicted by the MinneTAC agent and the real price during the games. The procedure was performed on the same way as with the experiments to determine the predictive performance of the MinneTAC agent with different values of the block weight horizon and τ parameter value in section 5.2.1. The RMSE was explained in section 5.1.3 on page 46. The experimental design was explained in section 5.1 on page 42.

7.4 The results

This section is ordered as follows. A discussion of the results with exponential variance calculation is in the first section. The results with block

weighting variance calculation are in the next section. A comparison between the exponential and block weighting variance calculation methods are in the last section.

7.4.1 Dynamic economic regime transitions with exponential variance calculation

Exponential variance calculation was used to determine the weights of each price predictor in the ensemble. A sigmoid and linear transition function were used for the transitions between the different game phases. The performance of the MinneTAC agent, with dynamic economic regime transitions, was compared with a version of the MinneTAC agent without economic regime transitions. The MinneTAC agent without dynamic economic regime showed a better predictive performance. This is contrary to the expectations. The dynamic economic regime probabilities were implemented to increase the performance of the MinneTAC agent. Future research has to identify why the performance lagged behind. A possible cause is the training data of the different game phases. The training data was obtained from games of TAC SCM competitions from 2005. Training data from recent TAC SCM competitions could influence the predictive performance of the MinneTAC agent with dynamic economic regime transition probabilities. Furthermore, the transition length between the different economic regime transition probabilities and the game phases could be varied. These experiments were out of the scope of this research.

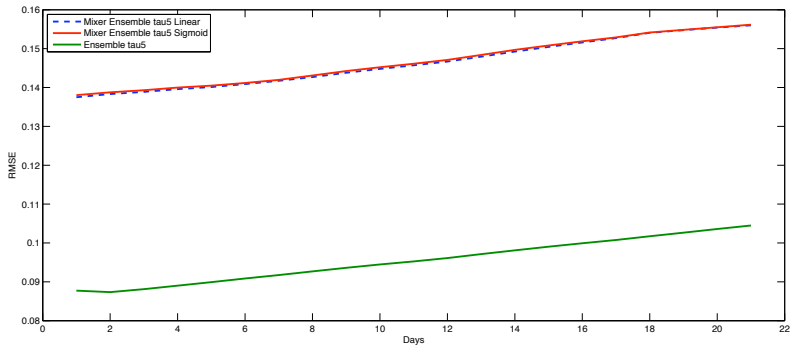

Figure 7.4: The outcomes from the experiments with dynamic economic regime transitions probabilities with exponential variance calculation.

The y-axis shows the average RMSE values of 40 TAC SCM games, with each version of the MinneTAC agent. The x-axis show the number of days the MinneTAC agent predicted the future prices. The predictive performance of the MinneTAC agent was measured with the RMSE.

An ANOVA analysis between the experiments of 7.4 is in the table below. The MinneTAC agent versions with dynamic regime transition probabilities had a significant difference in the predictive performance compared with the MinneTAC without dynamic regime probabilities.

Source	**SS**	**df**	**MS**	**F**	**Prob > F**
Columns	0.02754	1	0.02754	819.88	0
Error	0.00134	40	0.00003		
Total	0.002888	41			

7.4.2 Dynamic economic regime transitions with block weighting variance calculation

The experiments with dynamic economic regime transition with block weighting variance calculation are in 7.5. The MinneTAC agent with dynamic regime transition probabilities was not able to outperform the MinneTAC agent without dynamic economic regime transitions. This was in line with the results with exponential variance calculation.

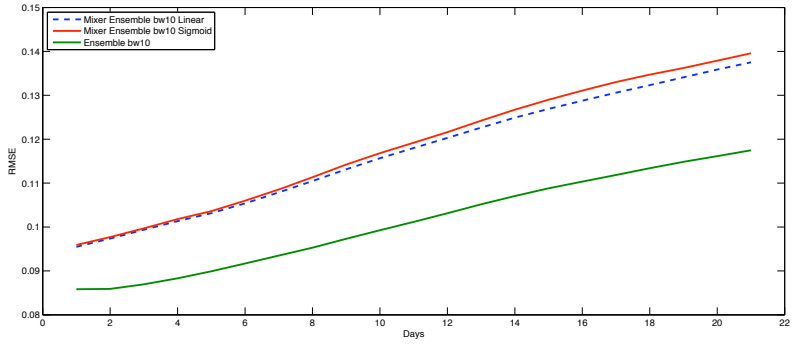

Figure 7.5: The outcomes from the experiments with dynamic economic regime transitions probabilities with block weighting variance calculation.

An ANOVA analysis between the experiments of 7.5 are in the table below. With block weighting variance calculation, the price prediction mechanism of the MinneTAC agent also showed a significantly less accurate predictive performance compared to the version of the MinneTAC agent without dynamic economic regime transitions.

Source	**SS**	**df**	**MS**	**F**	**Prob > F**
Columns	0.00272	1	0.00272	18.1	0.001
Error	0.006	40	0.00015		
Total	0.00872	41			

7.4.3 A comparison between block weighting and exponential variance calculation

7.6 shows the differences in the predictive performance of the MinneTAC agent, configured with linear economic regime transition probabilities with exponential and block weighting variance calculation.

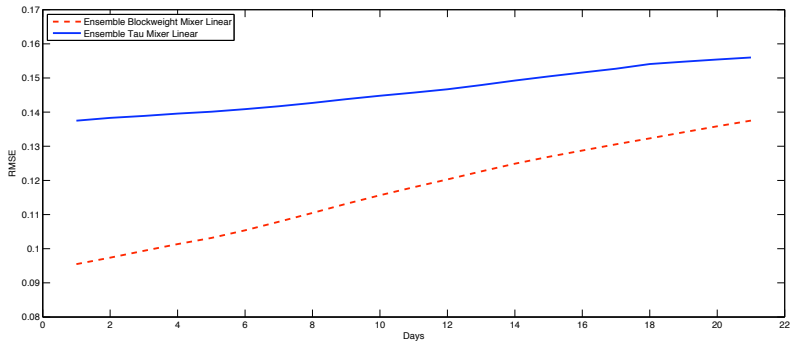

Figure 7.6:

The block weighting variance calculation method showed a better predictive than the exponential variance calculation method. This could mean the dynamic economic regime transition probabilities work better with block weighting variance calculation. Another option is the variance between experiments. Because 40 TAC SCM games for each experiment were performed, this option is not likely.

The table below shows a significant difference between the experiments with exponential and block weighting variance calculation, with linear economic regime probability transitions.

Source	**SS**	**df**	**MS**	**F**	**Prob > F**
Columns	0.00891	1	0.00891	79.9	0.00004
Error	0.00446	40	0.00011		
Total	0.01337	41			

An overview between the price predictions of the block weighting and exponential variance calculation method are in 7.7 to give an overview of the differences in the predictive performance. There can be concluded that there are no differences between the sigmoid and linear transition functions for exponential and block weighting variance calculation. The main finding is that there are no differences in the predictive performance of linear and sigmoid economic regime transition probabilities.

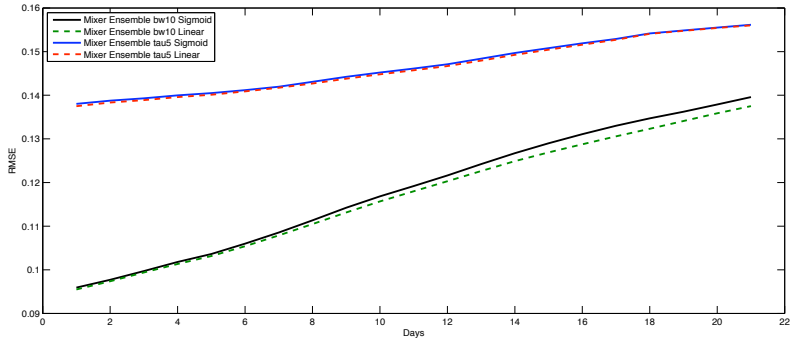

Figure 7.7: A comparison between the experiments with block weighting and exponential variance calculation

Chapter 8

Conclusion and future research

This research was conducted with the aim to improve the ensemble price prediction mechanism of the MinneTAC agent. The conclusion is divided in three parts. The main findings of improving ensemble price prediction in the MinneTAC agent are discussed in the first part. The outcomes of implementing dynamic economic regime transition probabilities are discussed in the second part. The vision of the authors is given in the last part.

8.1 Improving ensemble price prediction

The main outcome of this research is a better insight in the ensemble price prediction mechanism of the MinneTAC agent. The goal of ensemble price prediction is to combine multiple price prediction methods in such a way that the disadvantages of each price predictor are compensated. This should result in an ensemble price that is more accurate than the price predictions of the individual price predictors. The ensemble price prediction mechanism of the MinneTAC agent did not outperform the individual price predictors. This implies that the dynamic weighting mechanism of the MinneTAC agent was not able to optimally compensate the disadvantages of each price predictor. This research identified two causes why the performance of the ensemble price did not outperform best performing price predictors.

In the first place, the weighting mechanism has to be revised. Weights are assigned to each price predictor based the predictive performance. Price predictors, with a less accurate performance, are still assigned a substantial weight. This negatively influences the predictive performance of the ensemble. A voting system could provide a solution, in which the best performing price predictor is only used.

In the second place, a potential deficiency of distinctive properties between the price prediction methods in the ensemble of the MinneTAC agent could have preclude more accurate price predictions. Only the exponential-2c

adapter is not based on economic regimes and was added to improve short term ahead price predictions. The Markov, Markov-correction and exponential price predictor are all using economic regimes to predict the future prices. The idea of ensemble price prediction is to use price predictors with different characteristics. The ensemble could be extended with new price predictors without economic regimes. Support Vector Machines or Neural Networks are well suited to extend the ensemble of price predictors in the MinneTAC agent. The advance of these techniques is the capability the handle large non-linear relationships [11] [10].

The Markov-correction predictor could be removed from the ensemble. The Markov and Markov-correction predictor did not had any significant differences in the predictive performance of the ensemble. This means that the Markov-correction predictor could be removed without influencing the predictive performance of the MinneTAC agent.

The dynamic weighting mechanism of the MinneTAC agent assigns weights to each price predictor based on the predictive performance. The predictive performance is based on the variance in past price predictions. The variance could be calculated with an exponential and block weighting variance calculation method. The number of historical price predictions had only a very marginal influence on the predictive performance of the price predictions with block weighting variance calculation. The emphasis on the current observed variance compared to previously calculated variances had only a very modest influence in the predictive performance of the exponential variance calculation method.

Bootstrapping is a form of supervised learning and was implemented to further improve the predictive performance of the MinneTAC agent in the first part of TAC SCM games [11]. The results from bootstrapping the MinneTAC agent, with the optimal weights of each price predictor for the first game phase, did only had a very marginal increase in the predictive performance. The increase in the predictive performance was not statistically significant. This means that the differences between assigning equal weights to each price predictor or the weights calculated by the dynamic weighting mechanism for the first game phases did not had a significant difference. This leads to the question of how large the differences in the predictive performance are between equal weights or the weights of the dynamic weighting mechanism.

8.2 The implementation of dynamic economic regime transition probabilities

Dynamic economic regime transition probabilities, for each game phase, were implemented in the MinneTAC agent to further improve price predictions. The main goal of this research was to implement the mechanism to enable dynamic economic regime transition probabilities. The dynamic economic

regime transition probabilities did not result in an increase in the predictive performance of the MinneTAC agent. Three causes were identified that could explain the outcomes from the experiments.

In the first place, economic regime transition probabilities were calculated with old game data from previously held TAC SCM competitions. New training data could influence the performance of the MinneTAC agent.

In the second place, the optimal settings of the mechanism that changes the economic regime transition probabilities of each game phase could be further investigated. In the last place, a possibilities is that the differences between the three games phase are very marginal and that implementing dynamic economic regime transition probabilities will not increase the predictive performance of the MinneTAC agent.

8.3 Vision of the authors

The vision of the authors on the working of the price prediction mechanism of the MinneTAC agent is given in this section. The MinneTAC agent is one of the best performing TAC SCM agents during the recent competitions. Last year, the MinneTAC reached the third position. We tried to further improve the price prediction mechanism of the MinneTAC agent. Predicting the future prices is a crucial part in the TAC SCM competition. The agent that is able to predict the future market prices as accurately as possible is most likely to win the most customer orders. A major breakthrough in the predictive performance of the MinneTAC agent was not discovered. We are convinced that the predictive performance of this price prediction mechanism could not be significantly improved. We propose to further developed the price prediction mechanism with a strong focus on flexibility and simplicity. The price prediction mechanism of the MinneTAC agent is relatively complex compared to the price prediction mechanism of the most direct competitors. We propose an ensemble price prediction mechanism based on a voting mechanism instead of the complex system currently implemented. This means that less accurate price predictors are not used without the complex calculations performed by the dynamic weighting mechanism.

Bibliography

[1] P. Markillie, "The physical internet," *SURVEY of LOGISTICS, the Economist*, June 2006.

[2] E. G. Anderson and D. J. Morrice, "A simulation model to study the dynamics in a service-oriented supply chain," in *WSC '99: Proceedings of the 31st conference on Winter simulation*, (New York, NY, USA), pp. 742–748, ACM, 1999.

[3] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*. McGraw-Hill, 1990.

[4] J. Collins, W. Ketter, M. Gini, and A. Agovic, "Software architecture of the MinneTAC supply-chain trading agent," Tech. Rep. 08-031, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, MN, October 2008.

[5] Michael Wooldridge and Nicholas R. Jennings, "Intelligent agents: Theory and practice," *Knowledge Engineering Review*, vol. 10, no. 1, pp. 115–152, 1995.

[6] J. Collins, W. Ketter, and M. Gini, "Architectures for Agents in TAC SCM," in *AAAI Spring Symposium on Architectures for Intelligent Theory-Based Agents*, (Stanford University, Palo Alto, California), pp. 7–12, March 2008.

[7] J. Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne, and S. Jansonc, "TAC SCM Game Description." Website, 2008. Available online, https://www.sics.se/tac/page.php last visited 15 april 2008.

[8] K. C. Chatzidimitriou, A. L. Symeonidis, I. Kontogounis, and P. A. Mitkas, "Agent Mertacor: A robust design for dealing with uncertainty and variation in SCM environments.," *Expert Systems with Applications*, vol. 35, pp. 591–603, Oct. 2008.

[9] W. Ketter, J. Collins, M. Gini, A. Gupta, and P. Schrater, "Detecting and Forecasting Economic Regimes in Multi-Agent Automated Exchanges," *Decision Support Systems*, Forthcoming 2009.

[10] F. O. Karray and C. D. Silva, *Soft Computing and Intelligent System Design, theory tools and applications*. Addison Wesley, 2004.

[11] S. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach; 2nd Edition*. Prentice Hall, 2002. ISBN: 0137903952.

[12] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[13] C. Hung and J.-H. Chen, "A selective ensemble based on expected probabilities for bankruptcy prediction," *Expert Systems with Applications*, vol. 36, no. 5, p. 52975303, 2009.

[14] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 4, p. 123140, 1996.

[15] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, p. 119139, 1997.

[16] Z. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artificial Intelligence*, vol. 137, no. 2, p. 239263, 2002.

[17] J.-J. Guo and P. B. Luh, "Improving market clearing price prediction by using a committee machine of neural networks," *IEEE TRANSACTIONS ON POWER SYSTEMS*, vol. 19, no. 4, pp. 1867–1876, 2004.

[18] P. J. Adeodato, G. C. V. Adrian L. Arnaud, R. C. Cunha, and D. S. Monteiro, "Mlp ensembles improve long term prediction accuracy over single networks," *International Journal of Forecasting*, vol. In Press, Corrected Proof, no. 4, pp. –, 2009.

[19] L. Nanni and A. Luminia, "An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3028 – 3033, 2009.

[20] T. Ho, "The random subspace method for constructing decision forests," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 20, no. 8, pp. 832–844, 1998.

[21] H.-H. Won, M.-J. Kim, S. Kim, and J.-W. Kim, "Ensempro: An ensemble approach to predicting transcription start sites in human genomic dna sequences," *Genomics*, vol. 91, no. 3, pp. 259 – 266, 2008.

[22] Y. Kim, "Boosting and measuring the performance of ensembles for a successful database marketing," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2161 – 2176, 2009.

[23] M. Burdu and C. Wyplosz, *Macroeconomics, a European text*. Oxford university press, 2005.

[24] J. D. Hamilton, "A new approach to economic regime analysis of nonstationary time series and the business cycle," *Econometrica*, vol. 57, no. 2, pp. 357–384, 1989.

[25] A. J. Filardo, "Business-cycle phases and their transitional dynamics," *Journal of Business and Economic Statistics*, vol. 12, no. 3, pp. 299–314, 1994.

[26] W. Ketter, J. Collins, M. Gini, A. Gupta, and P. Schrater, "Detecting and Forecasting Economic Regimes in Multi-Agent Automated Exchanges," *Decision Support Systems*, vol. in publication, 2009.

[27] W.-H. Liu and Y.-L. Chyi, "A markov regime-switching model for the semiconductor industry cycles," *Economic Modelling*, vol. 23, no. 4, pp. 569 – 578, 2006.

[28] R. Ahrens, "Predicting recessions with interest rate spreads: a multicountry regime-switching analysis," *Journal of International Money and Finance*, vol. 21, no. 4, pp. 519 – 537, 2002.

[29] M. Guidolin and A. Timmermann, "Asset allocation under multivariate regime switching," *Journal of Economic Dynamics and Control*, vol. 31, no. 11, pp. 3503 – 3544, 2007.

[30] T. Angelidis and N. Tessaromatis, "Idiosyncratic risk matters! a regime switching approach," *International Review of Economics Finance*, vol. 18, no. 1, pp. 132 – 141, 2009.

[31] T. D. Mount, Y. Ning, and X. Cai, "Predicting price spikes in electricity markets using a regime-switching model with time-varying parameters," *Energy Economics*, vol. 28, no. 1, pp. 62 – 80, 2006.

[32] P. Stone, "Multiagent competitions and research: Lessons from RoboCup and TAC," in *RoboCup-2002: Robot Soccer World Cup VI* (G. A. Kaminka, P. U. Lima, and R. Rojas, eds.), pp. 224–237, Berlin: Springer Verlag, 2003.

[33] S. Nahmias, *Production and operations analysis*. McGraw-Hill, 2009.

[34] J. Collins, W. Ketter, M. Gini, and A. Agovic, "Software architecture of the MinneTAC supply-chain trading agent," Tech. Rep. 08-031, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, MN, October 2008.

[35] W. Ketter, *Identification and Prediction of Economic Regimes to Guide Decision Making in Multi-Agent Marketplaces*. PhD thesis, University of Minnesota, Twin-Cities, USA, January 2007.

[36] M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, and M. Tschantz, "Botticelli: A supply chain management agent designed to optimize under uncertainty," *ACM Trans. on Comp. Logic*, vol. 4, no. 3, pp. 29–37, 2004.

[37] M. Benisch, A. Sardinha, J. Andrews, and N. Sadeh, "Cmieux: adaptive strategies for competitive supply chain trading," in *ICEC '06: Proceedings of the 8th international conference on Electronic commerce*, (New York, NY, USA), pp. 47–58, ACM, 2006.

[38] P. W. Keller, F.-O. Duguay, and D. Precup, "Redagent-2003: An autonomous market-based supply-chain management agent," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1182 – 1189, July 2004.

[39] C. Kiekintveld, J. Miller, P. R. Jordan, and M. P. Wellman, "Forecasting market prices in a supply chain game," in *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, (New York, NY, USA), pp. 1–8, ACM, 2007.

[40] D. Pardoe and P. Stone, "An autonomous agent for supply chain management," in *Handbooks in Information Systems Series: Business Computing* (G. Adomavicius and A. Gupta, eds.), Elsevier, 2007.

[41] M. Stan, B. Stan, and A. M. Florea, "A dynamic strategy agent for supply chain management," in *SYNASC '06: Proceedings of the Eighth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, (Washington, DC, USA), pp. 227–232, IEEE Computer Society, 2006.

[42] W. Ketter, J. Collins, M. Gini, A. Gupta, and P. Schrater, "Pricing and resource allocation for intelligent trading agents using economic regimes," in *International Symposium of Information Systems*, (Hyderabad, India), pp. –, December 2007.

[43] G. Shmueli, N. Patel, and P. Bruce, *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner*. Wiley-Interscience, 2006.

[44] W. Ketter, "Dynamic Regime Identification and Prediction Based on Observed Behavior in Electronic Marketplaces," in *Proc. of the Twentieth National Conference on Artificial Intelligence*, (Pittsburgh), pp. 1646–1647, July 2005.

[45] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. Springer series in statistics, 2009.

[46] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*. Springer series in statistics, 2001.

[47] Hyoung-Joo and S. Cho, "Combining gaussian mixture models," *Intelligent Data Engineering and Automated Learning-IDEAL 2004*, vol. 5, no. 2, pp. 666–671, 2004.

[48] W. Ketter, J. Collins, M. Gini, A. Gupta, and P. Schrater, "Strategic sales management guided by economic regimes," in *Edited Volume of the 2nd Smart Business Network Initiative Discovery Event* (E. van Heck et al., ed.), Springer Verlag, 2006.

[49] W. Ketter, J. Collins, M. Gini, A. Gupta, and P. Schrater, "Detecting and forecasting economic regimes in automated exchanges," Tech. Rep. 07-007, University of Minnesota, Dept of Computer Science and Engineering, Minneapolis, MN 55455, 2007.

[50] W. Ketter, J. Collins, M. Gini, A. Gupta, and P. Schrater, "Ensemble Market Price Prediction of Trading Agents based on Economic Regimes." 2009.

[51] E. Sodomka, J. Collins, and M. Gini, "Efficient statistical methods for evaluating trading agent performance," in *Proc. of the Twenty-Second National Conference on Artificial Intelligence*, pp. 770–775, 2007.

[52] A. Gilat, *Matlab An Introduction With Applications*. John Wiley & Sons, 2004.

[53] D. Moore, G. McGabe, W. Duckworth, and S. Sclove, *The practice of business statistics*. Freeman and Company, 2003.

[54] R. Schuessler, "The gradual decline of cooperation: Endgame effects in evolutionary game theory," *Theory and Decision*, vol. 26, no. 2, pp. 133–155, 1989.

Appendix

BW/Days	1	2	3	4	5	6	7	8	9	10	11
5	0.088	0.0878	0.089	0.0905	0.0922	0.094	0.0959	0.0979	0.0999	0.1019	0.1039
	(0.0460)	(0.0444)	(0.0439)	(0.0433)	(0.0428)	(0.0425)	(0.0424)	(0.0426)	(0.0431)	(0.0436)	(0.0443)
10	0.0868	0.0866	0.0879	0.0894	0.0911	0.093	0.095	0.097	0.099	0.1009	0.1029
	(0.0451)	(0.0436)	(0.0432)	(0.0428)	(0.0424)	(0.0423)	(0.0424)	(0.0428)	(0.0432)	(0.0438)	(0.0445)
15	0.094	0.0935	0.0946	0.0959	0.0973	0.0989	0.1005	0.1022	0.104	0.1057	0.1073
	(0.0543)	(0.0527)	(0.0524)	(0.0519)	(0.0515)	(0.0511)	(0.0511)	(0.0512)	(0.0513)	(0.0516)	(0.0521)
20	0.085	0.0847	0.0864	0.0886	0.091	0.0934	0.0959	0.0985	0.1011	0.1037	0.1062
	(0.0455)	(0.0442)	(0.0439)	(0.0436)	(0.0436)	(0.0437)	(0.0442)	(0.0448)	(0.0456)	(0.0466)	(0.0476)
25	0.0891	0.0882	0.0896	0.0912	0.093	0.095	0.0968	0.0987	0.1006	0.1024	0.1042
	(0.0451)	(0.0430)	(0.0425)	(0.0419)	(0.0414)	(0.0412)	(0.0412)	(0.0414)	(0.0417)	(0.0424)	(0.0433)

BW/Days	12	13	14	15	16	17	18	19	20	21
5	0.1058	0.1078	0.1097	0.1115	0.1131	0.1146	0.116	0.1175	0.1189	0.1202
	(0.0449)	(0.0459)	(0.0469)	(0.0481)	(0.0492)	(0.0502)	(0.0512)	(0.0523)	(0.0533)	(0.0543)
10	0.1049	0.107	0.1089	0.1108	0.1124	0.1138	0.1152	0.1167	0.1182	0.1196
	(0.0453)	(0.0465)	(0.0479)	(0.0493)	(0.0507)	(0.0520)	(0.0535)	(0.0549)	(0.0563)	(0.0577)
15	0.109	0.1105	0.1119	0.1131	0.1143	0.1154	0.1164	0.1176	0.1187	0.1197
	(0.0527)	(0.0534)	(0.0540)	(0.0546)	(0.0550)	(0.0552)	(0.0557)	(0.0563)	(0.0569)	(0.0573)
20	0.1087	0.1113	0.1136	0.1158	0.1179	0.1199	0.1218	0.1235	0.1251	0.1266
	(0.0488)	(0.0503)	(0.0518)	(0.0534)	(0.0548)	(0.0561)	(0.0578)	(0.0593)	(0.0609)	(0.0622)
25	0.1061	0.1081	0.1101	0.1119	0.1136	0.1153	0.1169	0.1185	0.12	0.1215
	(0.0443)	(0.0458)	(0.0473)	(0.0488)	(0.0501)	(0.0513)	(0.0526)	(0.0541)	(0.0556)	(0.0571)

Table 8.1: Block weight RMSE and standard deviation

tau/Days	1	2	3	4	5	6	7	8	9	10	11
1	0.0994	0.0987	0.0995	0.1004	0.1013	0.1023	0.1033	0.1041	0.1049	0.1055	0.1062
	(0.055)	(0.0532)	(0.0527)	(0.0523)	(0.0518)	(0.0514)	(0.0512)	(0.0513)	(0.0515)	(0.0516)	(0.0519)
2	0.0965	0.0962	0.097	0.098	0.099	0.1001	0.1011	0.1022	0.1032	0.1042	0.1051
	(0.0499)	(0.0484)	(0.0479)	(0.0476)	(0.0473)	(0.0471)	(0.047)	(0.047)	(0.0472)	(0.0476)	(0.0481)
4	0.0918	0.0936	0.0945	0.0955	0.0965	0.0975	0.0985	0.0996	0.1006	0.1017	0.1026
	(0.045)	(0.0456)	(0.0453)	(0.045)	(0.0449)	(0.0448)	(0.0449)	(0.0452)	(0.0456)	(0.0461)	(0.0468)
5	0.0877	0.0874	0.0881	0.089	0.0899	0.0909	0.0917	0.0927	0.0936	0.0945	0.0953
	(0.0428)	(0.0402)	(0.0396)	(0.0391)	(0.0386)	(0.0383)	(0.0382)	(0.0382)	(0.0384)	(0.0387)	(0.0392)
7	0.087	0.0866	0.0875	0.0886	0.0896	0.0907	0.0918	0.0929	0.0939	0.0948	0.0957
	(0.0436)	(0.0414)	(0.0409)	(0.0405)	(0.0401)	(0.04)	(0.04)	(0.0403)	(0.0407)	(0.0412)	(0.0418)
10	0.0862	0.0859	0.087	0.0883	0.0896	0.091	0.0923	0.0936	0.0949	0.096	0.0971
	(0.0423)	(0.0395)	(0.0389)	(0.0384)	(0.0381)	(0.0379)	(0.0378)	(0.0378)	(0.038)	(0.0383)	(0.0387)

tau/Days	12	13	14	15	16	17	18	19	20	21
1	0.1069	0.1078	0.1086	0.1094	0.1103	0.1111	0.1122	0.1132	0.114	0.1148
	(0.0523)	(0.0533)	(0.0543)	(0.0554)	(0.0564)	(0.0575)	(0.059)	(0.0598)	(0.0608)	(0.0617)
2	0.1059	0.107	0.1081	0.1092	0.1103	0.1114	0.1126	0.1135	0.1143	0.115
	(0.0488)	(0.0501)	(0.0516)	(0.0528)	(0.054)	(0.0553)	(0.0568)	(0.0577)	(0.0586)	(0.0595)
4	0.1035	0.1045	0.1057	0.1069	0.108	0.1091	0.1103	0.1113	0.1122	0.113
	(0.0475)	(0.0488)	(0.05)	(0.0511)	(0.0522)	(0.0533)	(0.0552)	(0.0563)	(0.0574)	(0.0585)
5	0.0961	0.0971	0.0981	0.099	0.0999	0.1008	0.1017	0.1027	0.1036	0.1045
	(0.0397)	(0.0409)	(0.0422)	(0.0435)	(0.0451)	(0.0466)	(0.0488)	(0.0501)	(0.0515)	(0.0528)
7	0.0966	0.0975	0.0985	0.0995	0.1006	0.1015	0.1026	0.1036	0.1045	0.1054
	(0.0424)	(0.0438)	(0.0453)	(0.0467)	(0.0478)	(0.049)	(0.0508)	(0.0518)	(0.0528)	(0.0538)
10	0.0981	0.0994	0.1004	0.1014	0.1023	0.1031	0.104	0.1048	0.1055	0.1063
	(0.0392)	(0.0405)	(0.0418)	(0.043)	(0.0441)	(0.0454)	(0.0471)	(0.0483)	(0.0496)	(0.0508)

Table 8.2: TAU 5 RMSE and standard deviation

Exponential	Day 01	Day 02	Day 03	Day 04	Day 05	Day 06	Day 07	Day 08	Day 09	Day 10	Day 11
Product 1	0.169	0.174	0.147	0.129	0.116	0.101	0.096	0.092	0.086	0.084	0.084
Product 2	0.170	0.199	0.172	0.149	0.133	0.114	0.104	0.097	0.088	0.086	0.085
Product 3	0.209	0.227	0.196	0.174	0.156	0.141	0.135	0.126	0.120	0.115	0.112
Product 4	0.249	0.253	0.229	0.206	0.182	0.160	0.146	0.134	0.125	0.119	0.115
Product 5	0.143	0.204	0.178	0.159	0.143	0.128	0.121	0.117	0.111	0.110	0.111
Product 6	0.180	0.201	0.166	0.143	0.125	0.109	0.100	0.095	0.092	0.091	0.091
Product 7	0.148	0.224	0.195	0.171	0.153	0.136	0.128	0.120	0.112	0.109	0.107
Product 8	0.178	0.266	0.241	0.212	0.187	0.173	0.161	0.154	0.146	0.140	0.138
Product 9	0.179	0.180	0.154	0.133	0.116	0.099	0.091	0.087	0.083	0.083	0.084
Product 10	0.169	0.166	0.143	0.127	0.113	0.103	0.099	0.097	0.096	0.099	0.101
Product 11	0.172	0.190	0.167	0.143	0.126	0.112	0.104	0.099	0.091	0.091	0.094
Product 12	0.173	0.191	0.165	0.145	0.130	0.118	0.111	0.104	0.096	0.095	0.096
Product 13	0.227	0.266	0.239	0.212	0.194	0.177	0.166	0.159	0.153	0.151	0.149
Product 14	0.163	0.187	0.162	0.141	0.127	0.111	0.107	0.105	0.102	0.104	0.105
Product 15	0.213	0.265	0.232	0.207	0.187	0.174	0.166	0.160	0.154	0.150	0.148
Product 16	0.180	0.246	0.211	0.182	0.161	0.149	0.141	0.136	0.130	0.126	0.124

Exponential	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21
Product 1	0.079	0.077	0.076	0.074	0.073	0.074	0.074	0.074	0.075	0.078
Product 2	0.080	0.078	0.077	0.076	0.076	0.076	0.075	0.075	0.075	0.077
Product 3	0.110	0.109	0.109	0.108	0.108	0.109	0.108	0.108	0.110	0.113
Product 4	0.114	0.116	0.118	0.121	0.125	0.126	0.127	0.127	0.128	0.130
Product 5	0.108	0.106	0.106	0.105	0.105	0.105	0.104	0.105	0.105	0.107
Product 6	0.089	0.089	0.090	0.089	0.088	0.088	0.089	0.088	0.089	0.090
Product 7	0.103	0.102	0.102	0.100	0.099	0.099	0.100	0.101	0.101	0.105
Product 8	0.136	0.136	0.137	0.136	0.138	0.138	0.138	0.138	0.139	0.141
Product 9	0.081	0.082	0.083	0.083	0.085	0.086	0.087	0.087	0.088	0.089
Product 10	0.101	0.105	0.108	0.110	0.112	0.117	0.120	0.121	0.122	0.126
Product 11	0.093	0.095	0.096	0.096	0.095	0.097	0.097	0.096	0.097	0.100
Product 12	0.094	0.094	0.095	0.094	0.094	0.095	0.096	0.099	0.100	0.102
Product 13	0.147	0.145	0.144	0.143	0.144	0.145	0.143	0.141	0.141	0.141
Product 14	0.103	0.103	0.104	0.104	0.106	0.107	0.108	0.108	0.109	0.113
Product 15	0.146	0.146	0.145	0.143	0.144	0.143	0.142	0.141	0.141	0.142
Product 16	0.120	0.119	0.118	0.117	0.117	0.117	0.117	0.117	0.117	0.119

Table 8.3: Exponential predictor weights with exponential variance calculation

Markov	Day 01	Day 02	Day 03	Day 04	Day 05	Day 06	Day 07	Day 08	Day 09	Day 10	Day 11
Product 1	0.303	0.307	0.325	0.340	0.350	0.360	0.367	0.370	0.376	0.377	0.377
Product 2	0.317	0.312	0.329	0.345	0.357	0.369	0.376	0.378	0.385	0.385	0.386
Product 3	0.288	0.287	0.300	0.316	0.328	0.338	0.344	0.348	0.352	0.354	0.356
Product 4	0.261	0.266	0.277	0.293	0.307	0.322	0.331	0.338	0.344	0.347	0.347
Product 5	0.371	0.347	0.359	0.374	0.384	0.390	0.397	0.400	0.404	0.405	0.405
Product 6	0.362	0.356	0.377	0.390	0.400	0.408	0.412	0.415	0.418	0.419	0.420
Product 7	0.377	0.346	0.359	0.369	0.379	0.390	0.395	0.398	0.403	0.405	0.406
Product 8	0.367	0.330	0.342	0.357	0.371	0.379	0.384	0.386	0.389	0.391	0.390
Product 9	0.310	0.317	0.337	0.355	0.369	0.382	0.391	0.395	0.401	0.403	0.404
Product 10	0.327	0.335	0.350	0.364	0.374	0.382	0.389	0.392	0.393	0.396	0.395
Product 11	0.314	0.312	0.328	0.344	0.356	0.367	0.377	0.385	0.391	0.394	0.395
Product 12	0.315	0.311	0.329	0.343	0.353	0.364	0.371	0.378	0.383	0.386	0.388
Product 13	0.301	0.287	0.299	0.315	0.327	0.338	0.344	0.347	0.349	0.350	0.351
Product 14	0.353	0.347	0.361	0.376	0.385	0.399	0.404	0.406	0.409	0.409	0.407
Product 15	0.335	0.312	0.325	0.339	0.349	0.358	0.362	0.366	0.368	0.369	0.368
Product 16	0.358	0.330	0.345	0.361	0.373	0.382	0.387	0.389	0.390	0.390	0.390

Markov	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21
Product 1	0.381	0.383	0.382	0.383	0.382	0.381	0.380	0.379	0.378	0.376
Product 2	0.388	0.391	0.391	0.392	0.392	0.391	0.392	0.393	0.393	0.393
Product 3	0.358	0.360	0.361	0.362	0.363	0.364	0.364	0.366	0.365	0.364
Product 4	0.348	0.345	0.341	0.338	0.334	0.333	0.331	0.332	0.331	0.330
Product 5	0.408	0.410	0.410	0.411	0.411	0.411	0.412	0.412	0.412	0.411
Product 6	0.422	0.422	0.421	0.422	0.422	0.421	0.420	0.420	0.420	0.419
Product 7	0.410	0.412	0.412	0.414	0.414	0.413	0.412	0.412	0.411	0.409
Product 8	0.390	0.391	0.390	0.390	0.389	0.388	0.388	0.388	0.387	0.386
Product 9	0.406	0.407	0.406	0.406	0.406	0.406	0.406	0.406	0.406	0.406
Product 10	0.396	0.395	0.393	0.392	0.391	0.389	0.388	0.387	0.386	0.384
Product 11	0.398	0.398	0.397	0.398	0.398	0.397	0.397	0.398	0.399	0.398
Product 12	0.392	0.393	0.393	0.395	0.396	0.396	0.396	0.396	0.396	0.396
Product 13	0.354	0.356	0.357	0.359	0.360	0.361	0.363	0.366	0.367	0.368
Product 14	0.409	0.409	0.407	0.408	0.407	0.406	0.406	0.405	0.405	0.404
Product 15	0.369	0.369	0.369	0.370	0.369	0.369	0.370	0.371	0.371	0.371
Product 16	0.392	0.393	0.393	0.394	0.393	0.392	0.393	0.393	0.392	0.391

Table 8.4: Markov predictor weights with exponential variance calculation

Markov-cor.	Day 01	Day 02	Day 03	Day 04	Day 05	Day 06	Day 07	Day 08	Day 09	Day 10	Day 11
Product 1	0.367	0.368	0.379	0.387	0.391	0.397	0.397	0.394	0.392	0.389	0.387
Product 2	0.367	0.352	0.361	0.372	0.378	0.386	0.388	0.391	0.395	0.396	0.397
Product 3	0.318	0.310	0.321	0.334	0.342	0.351	0.352	0.355	0.358	0.361	0.362
Product 4	0.284	0.285	0.294	0.308	0.322	0.336	0.345	0.351	0.355	0.358	0.358
Product 5	0.431	0.396	0.404	0.407	0.413	0.422	0.421	0.419	0.420	0.418	0.415
Product 6	0.377	0.369	0.383	0.394	0.403	0.412	0.417	0.419	0.419	0.417	0.416
Product 7	0.418	0.377	0.386	0.398	0.406	0.412	0.415	0.417	0.420	0.419	0.418
Product 8	0.379	0.336	0.343	0.356	0.368	0.375	0.380	0.383	0.386	0.389	0.389
Product 9	0.375	0.376	0.385	0.395	0.401	0.408	0.409	0.409	0.407	0.404	0.402
Product 10	0.374	0.377	0.386	0.393	0.400	0.405	0.404	0.403	0.402	0.397	0.394
Product 11	0.380	0.371	0.378	0.388	0.396	0.403	0.404	0.402	0.402	0.399	0.395
Product 12	0.377	0.366	0.373	0.381	0.389	0.393	0.393	0.393	0.395	0.394	0.392
Product 13	0.326	0.309	0.317	0.331	0.340	0.350	0.355	0.358	0.361	0.361	0.361
Product 14	0.394	0.379	0.387	0.394	0.401	0.406	0.405	0.403	0.401	0.398	0.397
Product 15	0.337	0.313	0.324	0.336	0.346	0.354	0.358	0.360	0.363	0.364	0.365
Product 16	0.372	0.339	0.353	0.366	0.376	0.381	0.383	0.383	0.386	0.388	0.388

Markov-cor.	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21
Product 1	0.389	0.390	0.390	0.392	0.392	0.392	0.391	0.391	0.390	0.387
Product 2	0.401	0.401	0.400	0.400	0.399	0.399	0.399	0.399	0.399	0.398
Product 3	0.364	0.366	0.366	0.368	0.368	0.368	0.369	0.371	0.370	0.369
Product 4	0.357	0.353	0.348	0.343	0.339	0.338	0.336	0.337	0.336	0.335
Product 5	0.415	0.415	0.415	0.416	0.416	0.416	0.416	0.415	0.414	0.412
Product 6	0.417	0.418	0.418	0.419	0.420	0.421	0.422	0.423	0.423	0.422
Product 7	0.419	0.419	0.418	0.419	0.420	0.419	0.418	0.418	0.417	0.414
Product 8	0.391	0.391	0.390	0.390	0.389	0.389	0.389	0.389	0.387	0.386
Product 9	0.402	0.402	0.400	0.400	0.400	0.399	0.399	0.400	0.400	0.400
Product 10	0.393	0.390	0.387	0.386	0.384	0.381	0.380	0.379	0.379	0.376
Product 11	0.395	0.393	0.392	0.393	0.395	0.395	0.396	0.397	0.397	0.395
Product 12	0.392	0.391	0.390	0.391	0.392	0.391	0.392	0.390	0.389	0.389
Product 13	0.363	0.364	0.365	0.366	0.365	0.366	0.368	0.369	0.370	0.371
Product 14	0.397	0.397	0.397	0.397	0.397	0.398	0.398	0.399	0.398	0.397
Product 15	0.367	0.368	0.368	0.369	0.369	0.370	0.371	0.372	0.371	0.372
Product 16	0.392	0.393	0.393	0.393	0.394	0.394	0.394	0.394	0.395	0.394

Table 8.5: Markov-cor. predictor weights with exp. variance calculation

Exp-2c	Day 01	Day 02	Day 03	Day 04	Day 05	Day 06	Day 07	Day 08	Day 09	Day 10	Day 11
Product 1	0.161	0.151	0.149	0.144	0.143	0.141	0.140	0.144	0.146	0.149	0.152
Product 2	0.146	0.137	0.138	0.135	0.133	0.132	0.132	0.133	0.133	0.132	0.132
Product 3	0.186	0.177	0.183	0.177	0.175	0.171	0.170	0.170	0.170	0.171	0.170
Product 4	0.206	0.197	0.200	0.194	0.188	0.183	0.179	0.178	0.176	0.177	0.180
Product 5	0.055	0.053	0.059	0.059	0.060	0.060	0.062	0.064	0.066	0.067	0.069
Product 6	0.081	0.075	0.074	0.073	0.072	0.071	0.071	0.072	0.072	0.073	0.073
Product 7	0.057	0.054	0.060	0.062	0.062	0.061	0.063	0.065	0.066	0.067	0.068
Product 8	0.077	0.068	0.075	0.075	0.074	0.074	0.075	0.077	0.079	0.081	0.083
Product 9	0.136	0.127	0.124	0.118	0.114	0.111	0.109	0.109	0.109	0.110	0.111
Product 10	0.130	0.123	0.121	0.117	0.113	0.110	0.109	0.109	0.109	0.109	0.110
Product 11	0.135	0.128	0.128	0.124	0.122	0.118	0.115	0.115	0.115	0.116	0.116
Product 12	0.135	0.131	0.134	0.131	0.129	0.125	0.124	0.125	0.126	0.125	0.124
Product 13	0.147	0.138	0.146	0.142	0.139	0.135	0.135	0.136	0.137	0.139	0.139
Product 14	0.090	0.087	0.090	0.089	0.087	0.084	0.084	0.086	0.088	0.090	0.091
Product 15	0.116	0.110	0.119	0.118	0.118	0.114	0.114	0.115	0.116	0.116	0.118
Product 16	0.091	0.085	0.091	0.091	0.090	0.089	0.090	0.092	0.094	0.096	0.097

Exp-2c	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21
Product 1	0.152	0.151	0.153	0.152	0.152	0.153	0.155	0.156	0.158	0.159
Product 2	0.131	0.130	0.132	0.133	0.133	0.134	0.134	0.133	0.133	0.133
Product 3	0.167	0.165	0.165	0.163	0.161	0.159	0.158	0.156	0.156	0.155
Product 4	0.182	0.186	0.193	0.198	0.202	0.204	0.205	0.204	0.205	0.205
Product 5	0.069	0.068	0.069	0.068	0.068	0.068	0.068	0.068	0.069	0.070
Product 6	0.072	0.071	0.071	0.071	0.070	0.069	0.069	0.068	0.069	0.069
Product 7	0.068	0.067	0.068	0.068	0.068	0.069	0.070	0.070	0.071	0.073
Product 8	0.083	0.083	0.084	0.084	0.084	0.085	0.086	0.086	0.087	0.088
Product 9	0.110	0.109	0.110	0.110	0.109	0.108	0.108	0.107	0.106	0.106
Product 10	0.110	0.111	0.112	0.113	0.113	0.113	0.113	0.113	0.113	0.113
Product 11	0.115	0.114	0.115	0.114	0.113	0.111	0.110	0.109	0.108	0.107
Product 12	0.123	0.122	0.122	0.121	0.119	0.118	0.116	0.115	0.115	0.114
Product 13	0.136	0.135	0.135	0.132	0.131	0.129	0.126	0.124	0.122	0.120
Product 14	0.091	0.091	0.092	0.091	0.089	0.089	0.089	0.088	0.088	0.087
Product 15	0.117	0.117	0.118	0.118	0.118	0.118	0.118	0.117	0.117	0.116
Product 16	0.096	0.096	0.096	0.096	0.096	0.096	0.096	0.096	0.097	0.097

Table 8.6: Exp-2c predictor weights with exponential variance calculation

Exponential	Day 01	Day 02	Day 03	Day 04	Day 05	Day 06	Day 07	Day 08	Day 09	Day 10	Day 11
Product 1	0.236	0.245	0.252	0.261	0.268	0.274	0.279	0.282	0.284	0.289	0.294
Product 2	0.242	0.250	0.258	0.265	0.273	0.281	0.284	0.287	0.291	0.294	0.294
Product 3	0.233	0.249	0.258	0.269	0.278	0.286	0.291	0.293	0.294	0.296	0.298
Product 4	0.226	0.238	0.248	0.257	0.268	0.274	0.282	0.287	0.292	0.296	0.298
Product 5	0.355	0.330	0.333	0.336	0.338	0.341	0.342	0.340	0.342	0.343	0.342
Product 6	0.359	0.338	0.343	0.345	0.346	0.347	0.345	0.346	0.349	0.351	0.352
Product 7	0.380	0.349	0.355	0.362	0.365	0.368	0.369	0.369	0.370	0.371	0.371
Product 8	0.332	0.306	0.309	0.313	0.317	0.319	0.321	0.321	0.323	0.325	0.326
Product 9	0.316	0.314	0.320	0.325	0.328	0.332	0.335	0.337	0.338	0.338	0.338
Product 10	0.289	0.301	0.302	0.304	0.304	0.304	0.304	0.304	0.305	0.305	0.304
Product 11	0.339	0.321	0.327	0.331	0.336	0.339	0.341	0.342	0.343	0.343	0.341
Product 12	0.328	0.315	0.321	0.326	0.329	0.333	0.337	0.339	0.340	0.339	0.339
Product 13	0.293	0.279	0.284	0.292	0.302	0.307	0.312	0.313	0.314	0.314	0.313
Product 14	0.261	0.276	0.278	0.279	0.281	0.279	0.279	0.276	0.274	0.274	0.272
Product 15	0.356	0.315	0.318	0.324	0.328	0.330	0.329	0.329	0.327	0.325	0.324
Product 16	0.352	0.316	0.321	0.325	0.327	0.329	0.329	0.327	0.326	0.324	0.321

Exponential	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21
Product 1	0.298	0.300	0.302	0.303	0.303	0.303	0.303	0.303	0.303	0.302
Product 2	0.297	0.299	0.300	0.300	0.298	0.297	0.296	0.295	0.295	0.294
Product 3	0.299	0.299	0.299	0.298	0.297	0.295	0.295	0.296	0.296	0.295
Product 4	0.300	0.300	0.298	0.296	0.295	0.295	0.297	0.299	0.300	0.301
Product 5	0.344	0.346	0.347	0.350	0.352	0.352	0.352	0.352	0.352	0.352
Product 6	0.354	0.356	0.357	0.360	0.362	0.362	0.360	0.361	0.362	0.361
Product 7	0.373	0.376	0.376	0.377	0.379	0.380	0.380	0.380	0.380	0.379
Product 8	0.328	0.330	0.333	0.336	0.337	0.337	0.338	0.338	0.339	0.339
Product 9	0.337	0.339	0.340	0.341	0.340	0.340	0.339	0.339	0.341	0.341
Product 10	0.304	0.304	0.304	0.304	0.304	0.304	0.304	0.305	0.306	0.306
Product 11	0.339	0.338	0.338	0.337	0.337	0.336	0.335	0.335	0.334	0.333
Product 12	0.338	0.338	0.338	0.337	0.337	0.337	0.337	0.338	0.340	0.340
Product 13	0.312	0.312	0.313	0.314	0.315	0.317	0.318	0.318	0.319	0.320
Product 14	0.271	0.271	0.270	0.271	0.273	0.273	0.273	0.274	0.274	0.274
Product 15	0.326	0.325	0.323	0.322	0.321	0.322	0.322	0.322	0.322	0.323
Product 16	0.320	0.320	0.321	0.322	0.324	0.326	0.324	0.323	0.326	0.327

Table 8.7: Exponential predictor weights with block weighting

Markov	Day 01	Day 02	Day 03	Day 04	Day 05	Day 06	Day 07	Day 08	Day 09	Day 10	Day 11
Product 1	0.259	0.264	0.271	0.277	0.281	0.282	0.284	0.285	0.291	0.297	0.302
Product 2	0.232	0.237	0.241	0.249	0.256	0.260	0.263	0.266	0.271	0.275	0.279
Product 3	0.250	0.262	0.271	0.281	0.286	0.289	0.290	0.290	0.291	0.293	0.294
Product 4	0.255	0.266	0.274	0.278	0.285	0.289	0.295	0.298	0.300	0.303	0.306
Product 5	0.354	0.330	0.327	0.327	0.329	0.331	0.332	0.329	0.332	0.334	0.333
Product 6	0.339	0.314	0.316	0.320	0.325	0.329	0.327	0.330	0.336	0.340	0.344
Product 7	0.350	0.316	0.323	0.332	0.336	0.343	0.344	0.344	0.349	0.352	0.354
Product 8	0.292	0.269	0.273	0.280	0.290	0.298	0.300	0.299	0.301	0.306	0.311
Product 9	0.319	0.315	0.319	0.325	0.329	0.333	0.336	0.339	0.341	0.340	0.339
Product 10	0.289	0.302	0.302	0.304	0.304	0.304	0.304	0.303	0.306	0.305	0.303
Product 11	0.344	0.325	0.329	0.334	0.338	0.342	0.345	0.346	0.346	0.344	0.343
Product 12	0.342	0.326	0.326	0.331	0.334	0.336	0.336	0.338	0.341	0.341	0.340
Product 13	0.287	0.270	0.271	0.280	0.290	0.298	0.303	0.306	0.306	0.306	0.305
Product 14	0.264	0.275	0.275	0.274	0.273	0.272	0.270	0.266	0.265	0.265	0.264
Product 15	0.312	0.276	0.282	0.290	0.300	0.305	0.308	0.310	0.311	0.311	0.312
Product 16	0.312	0.277	0.279	0.286	0.292	0.297	0.300	0.303	0.306	0.308	0.307

Markov	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21
Product 1	0.307	0.309	0.310	0.311	0.309	0.308	0.307	0.307	0.306	0.305
Product 2	0.285	0.289	0.292	0.293	0.292	0.291	0.291	0.291	0.291	0.291
Product 3	0.293	0.294	0.293	0.292	0.290	0.287	0.287	0.289	0.291	0.292
Product 4	0.308	0.307	0.305	0.302	0.300	0.302	0.303	0.305	0.306	0.306
Product 5	0.335	0.338	0.341	0.345	0.348	0.348	0.348	0.349	0.349	0.349
Product 6	0.348	0.352	0.353	0.358	0.361	0.361	0.359	0.361	0.362	0.361
Product 7	0.359	0.367	0.368	0.371	0.376	0.376	0.377	0.377	0.378	0.377
Product 8	0.317	0.321	0.326	0.329	0.331	0.332	0.333	0.333	0.334	0.334
Product 9	0.339	0.340	0.342	0.342	0.341	0.340	0.340	0.339	0.340	0.340
Product 10	0.304	0.305	0.305	0.304	0.304	0.304	0.304	0.304	0.305	0.306
Product 11	0.343	0.342	0.342	0.342	0.341	0.339	0.337	0.337	0.336	0.334
Product 12	0.340	0.342	0.341	0.339	0.339	0.339	0.339	0.340	0.342	0.341
Product 13	0.303	0.304	0.308	0.310	0.312	0.314	0.315	0.316	0.316	0.317
Product 14	0.264	0.265	0.265	0.267	0.269	0.271	0.271	0.273	0.273	0.273
Product 15	0.313	0.314	0.314	0.314	0.313	0.316	0.317	0.317	0.318	0.318
Product 16	0.308	0.309	0.313	0.315	0.320	0.322	0.321	0.321	0.324	0.325

Table 8.8: Markov predictor weights with block weighting

Markov-cor	Day 01	Day 02	Day 03	Day 04	Day 05	Day 06	Day 07	Day 08	Day 09	Day 10	Day 11
Product 1	0.218	0.221	0.213	0.202	0.196	0.193	0.193	0.189	0.182	0.177	0.172
Product 2	0.229	0.228	0.222	0.213	0.204	0.199	0.194	0.187	0.184	0.185	0.184
Product 3	0.214	0.207	0.193	0.182	0.177	0.171	0.168	0.167	0.166	0.166	0.169
Product 4	0.217	0.214	0.202	0.194	0.181	0.174	0.167	0.162	0.161	0.158	0.156
Product 5	0.257	0.305	0.305	0.300	0.294	0.286	0.283	0.287	0.279	0.275	0.276
Product 6	0.273	0.320	0.312	0.302	0.295	0.288	0.290	0.284	0.272	0.264	0.258
Product 7	0.248	0.315	0.301	0.284	0.275	0.264	0.259	0.258	0.251	0.245	0.241
Product 8	0.220	0.282	0.274	0.264	0.250	0.242	0.242	0.243	0.237	0.231	0.226
Product 9	0.273	0.282	0.272	0.257	0.248	0.237	0.228	0.220	0.214	0.212	0.210
Product 10	0.319	0.288	0.285	0.277	0.274	0.271	0.267	0.265	0.259	0.258	0.258
Product 11	0.234	0.277	0.266	0.254	0.244	0.233	0.226	0.223	0.220	0.220	0.219
Product 12	0.245	0.278	0.271	0.257	0.247	0.237	0.230	0.222	0.214	0.213	0.213
Product 13	0.228	0.272	0.263	0.245	0.226	0.211	0.203	0.198	0.197	0.198	0.200
Product 14	0.278	0.242	0.237	0.234	0.231	0.228	0.233	0.237	0.239	0.240	0.243
Product 15	0.192	0.287	0.277	0.259	0.241	0.229	0.224	0.217	0.215	0.214	0.210
Product 16	0.206	0.294	0.284	0.270	0.261	0.249	0.242	0.236	0.232	0.231	0.233

Markov-cor	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21
Product 1	0.164	0.159	0.158	0.155	0.154	0.152	0.151	0.150	0.150	0.149
Product 2	0.177	0.172	0.169	0.169	0.170	0.170	0.169	0.168	0.169	0.170
Product 3	0.169	0.166	0.165	0.166	0.167	0.168	0.167	0.167	0.166	0.168
Product 4	0.157	0.160	0.164	0.166	0.169	0.168	0.168	0.167	0.170	0.169
Product 5	0.270	0.263	0.256	0.247	0.240	0.239	0.236	0.234	0.233	0.231
Product 6	0.249	0.240	0.236	0.227	0.219	0.218	0.219	0.215	0.211	0.211
Product 7	0.232	0.219	0.217	0.211	0.203	0.200	0.198	0.197	0.195	0.196
Product 8	0.220	0.213	0.207	0.202	0.201	0.200	0.198	0.199	0.198	0.198
Product 9	0.209	0.203	0.198	0.195	0.196	0.196	0.195	0.197	0.194	0.194
Product 10	0.256	0.253	0.252	0.252	0.252	0.251	0.249	0.250	0.247	0.246
Product 11	0.221	0.220	0.219	0.218	0.218	0.220	0.222	0.220	0.221	0.223
Product 12	0.214	0.210	0.210	0.212	0.212	0.210	0.209	0.207	0.203	0.204
Product 13	0.202	0.199	0.191	0.187	0.185	0.184	0.184	0.186	0.186	0.185
Product 14	0.245	0.243	0.243	0.240	0.238	0.237	0.241	0.240	0.241	0.245
Product 15	0.203	0.202	0.201	0.203	0.203	0.199	0.199	0.201	0.200	0.199
Product 16	0.231	0.228	0.221	0.217	0.209	0.204	0.207	0.210	0.204	0.200

Table 8.9: Markov-correction predictor weights with block weighting

Exp-2c	Day 01	Day 02	Day 03	Day 04	Day 05	Day 06	Day 07	Day 08	Day 09	Day 10	Day 11
Product 1	0.287	0.270	0.265	0.260	0.255	0.251	0.244	0.244	0.243	0.238	0.233
Product 2	0.297	0.285	0.279	0.274	0.268	0.260	0.259	0.261	0.254	0.247	0.243
Product 3	0.304	0.282	0.278	0.269	0.259	0.254	0.250	0.251	0.250	0.245	0.240
Product 4	0.302	0.282	0.276	0.270	0.266	0.263	0.257	0.253	0.248	0.243	0.240
Product 5	0.035	0.034	0.035	0.037	0.039	0.042	0.044	0.045	0.047	0.048	0.050
Product 6	0.029	0.028	0.030	0.032	0.035	0.037	0.038	0.040	0.043	0.045	0.047
Product 7	0.022	0.020	0.021	0.023	0.024	0.026	0.027	0.029	0.030	0.032	0.034
Product 8	0.157	0.143	0.143	0.143	0.143	0.141	0.137	0.138	0.139	0.138	0.137
Product 9	0.092	0.089	0.090	0.093	0.095	0.098	0.101	0.105	0.108	0.111	0.114
Product 10	0.104	0.109	0.111	0.116	0.118	0.121	0.125	0.128	0.131	0.133	0.135
Product 11	0.083	0.077	0.078	0.081	0.083	0.086	0.088	0.090	0.092	0.094	0.096
Product 12	0.084	0.081	0.083	0.087	0.090	0.094	0.098	0.102	0.105	0.107	0.108
Product 13	0.192	0.180	0.182	0.183	0.182	0.184	0.182	0.183	0.182	0.182	0.182
Product 14	0.197	0.207	0.211	0.214	0.216	0.221	0.218	0.221	0.222	0.222	0.220
Product 15	0.141	0.122	0.124	0.127	0.132	0.136	0.139	0.144	0.147	0.149	0.155
Product 16	0.130	0.114	0.117	0.119	0.120	0.125	0.129	0.134	0.136	0.137	0.139

Exp-2c	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21
Product 1	0.231	0.231	0.231	0.231	0.234	0.237	0.239	0.239	0.241	0.244
Product 2	0.241	0.240	0.239	0.238	0.240	0.242	0.244	0.245	0.245	0.245
Product 3	0.239	0.241	0.243	0.245	0.246	0.250	0.251	0.249	0.247	0.246
Product 4	0.235	0.233	0.233	0.236	0.236	0.235	0.232	0.229	0.225	0.224
Product 5	0.051	0.054	0.056	0.058	0.060	0.062	0.064	0.065	0.066	0.068
Product 6	0.049	0.051	0.054	0.056	0.058	0.060	0.062	0.064	0.065	0.067
Product 7	0.036	0.038	0.039	0.041	0.042	0.044	0.045	0.046	0.048	0.049
Product 8	0.136	0.135	0.134	0.133	0.132	0.131	0.131	0.130	0.130	0.130
Product 9	0.116	0.118	0.121	0.122	0.123	0.124	0.125	0.125	0.125	0.125
Product 10	0.137	0.138	0.139	0.140	0.141	0.141	0.143	0.142	0.142	0.143
Product 11	0.098	0.100	0.101	0.103	0.104	0.105	0.107	0.108	0.109	0.109
Product 12	0.109	0.111	0.112	0.112	0.112	0.114	0.115	0.115	0.116	0.116
Product 13	0.183	0.185	0.188	0.189	0.189	0.186	0.184	0.180	0.178	0.179
Product 14	0.220	0.221	0.222	0.222	0.221	0.219	0.215	0.214	0.212	0.208
Product 15	0.158	0.160	0.162	0.163	0.163	0.163	0.162	0.160	0.160	0.160
Product 16	0.141	0.143	0.146	0.147	0.148	0.148	0.148	0.146	0.147	0.148

Table 8.10: Exponential-2c predictor weights with block weighting method

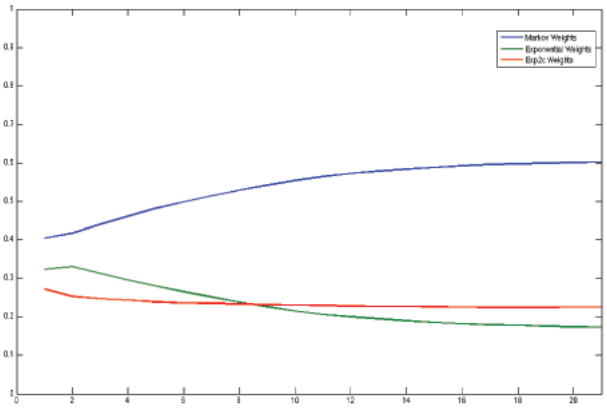

Figure 8.1: The weights for product 1 with block weighting variance calculation.

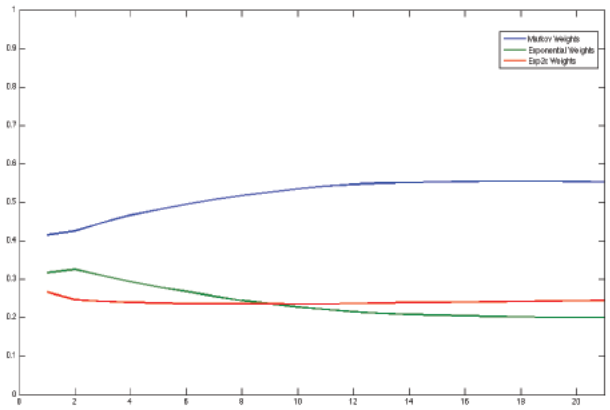

Figure 8.2: The weights for product 2 with block weighting variance calculation.

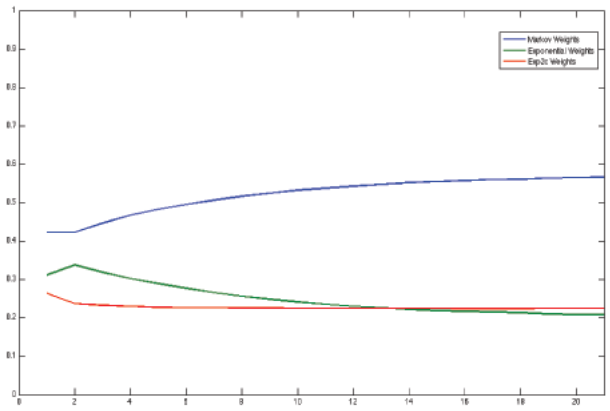

Figure 8.3: The weights for product 3 with block weighting variance calculation.

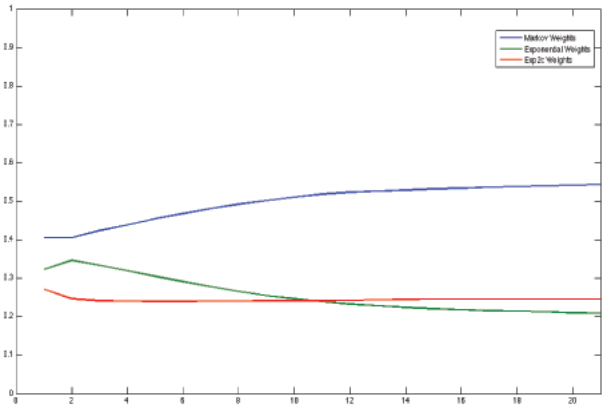

Figure 8.4: The weights for product 4 with block weighting variance calculation.

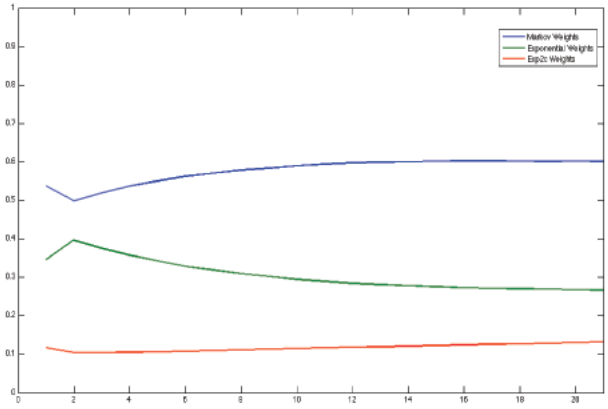

Figure 8.5: The weights for product 5 with block weighting variance calculation.

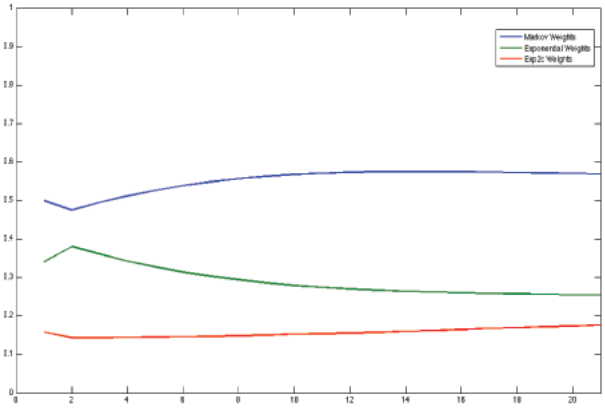

Figure 8.6: The weights for product 6 with block weighting variance calculation.

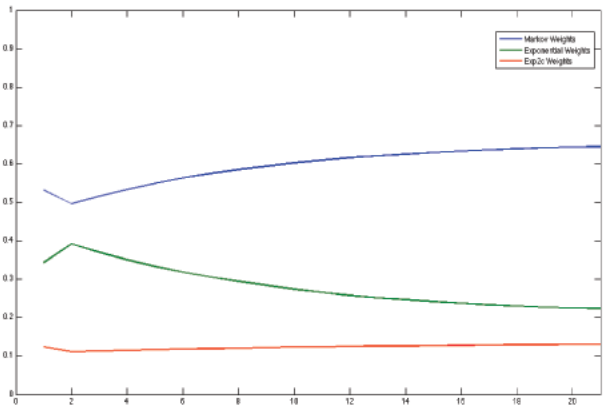

Figure 8.7: The weights for product 7 with block weighting variance calculation.

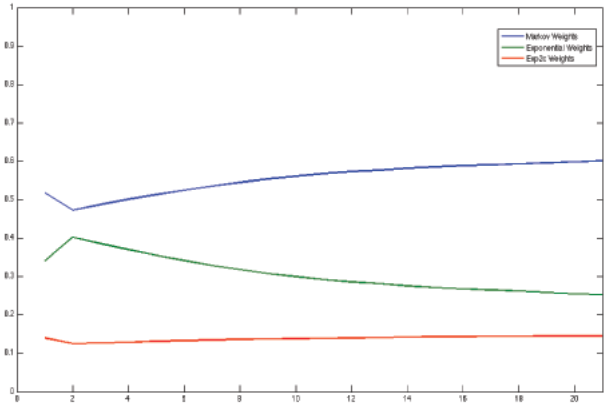

Figure 8.8: The weights for product 8 with block weighting variance calculation.

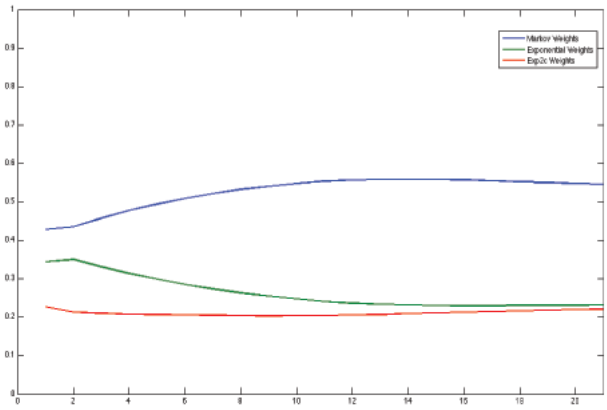

Figure 8.9: The weights for product 9 with block weighting variance calculation.

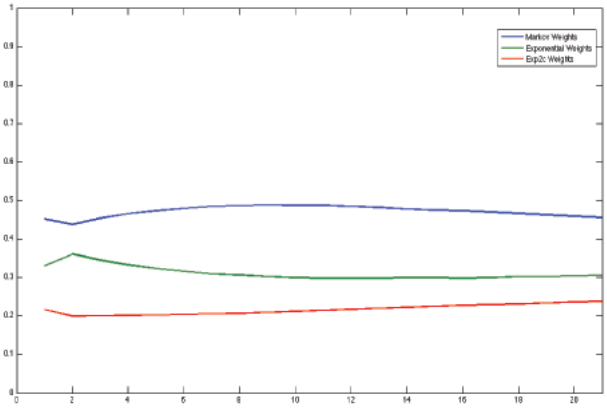

Figure 8.10: The weights for product 10 with block weighting variance calculation.

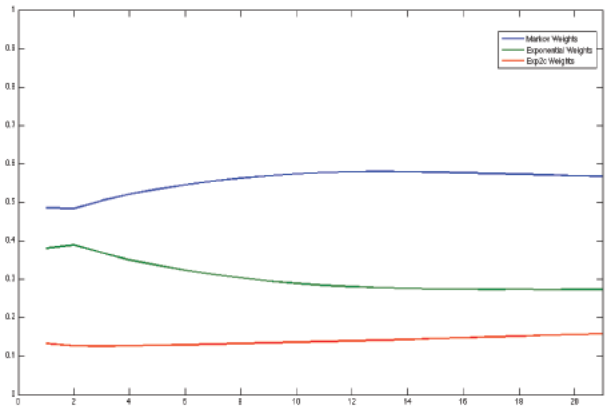

Figure 8.11: The weights for product 11 with block weighting variance calculation.

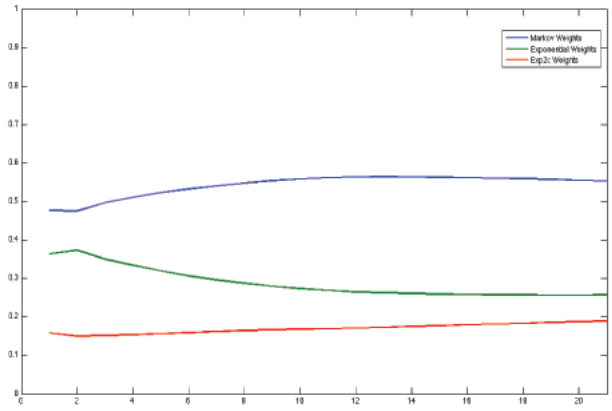

Figure 8.12: The weights for product 12 with block weighting variance calculation.

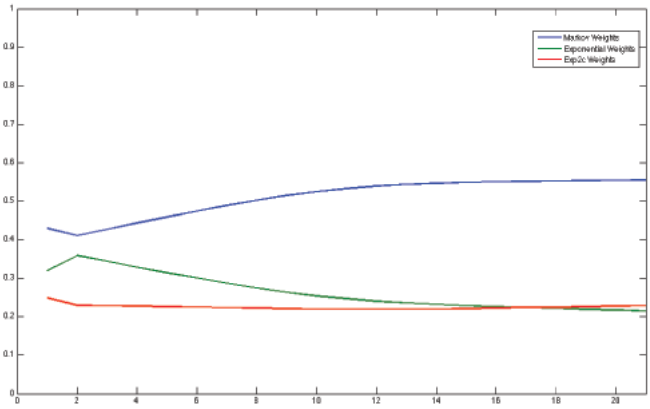

Figure 8.13: The weights for product 13 with block weighting variance calculation.

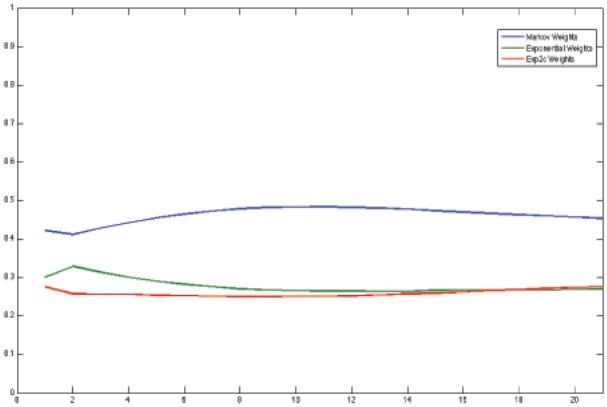

Figure 8.14: The weights for product 14 with block weighting variance calculation.

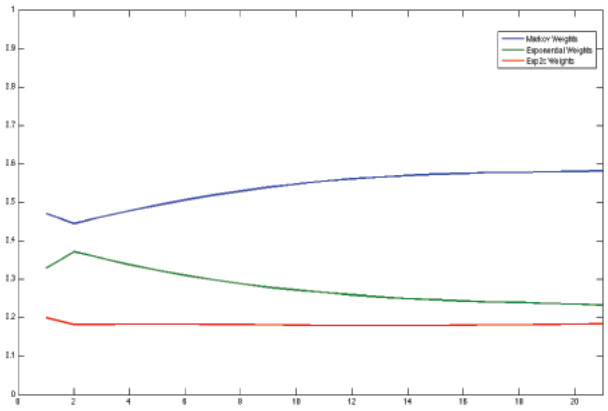

Figure 8.15: The weights for product 15 with block weighting variance calculation.

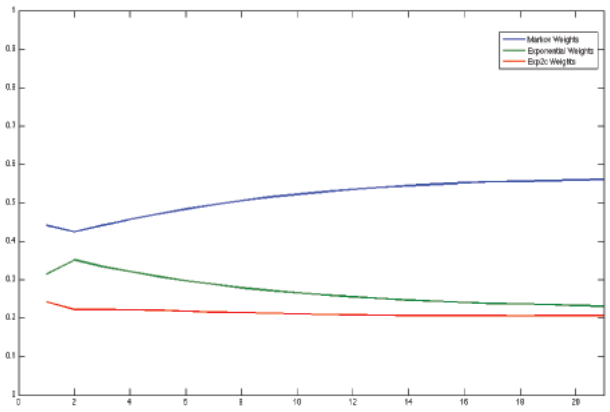

Figure 8.16: The weights for product 16 with block weighting variance calculation.

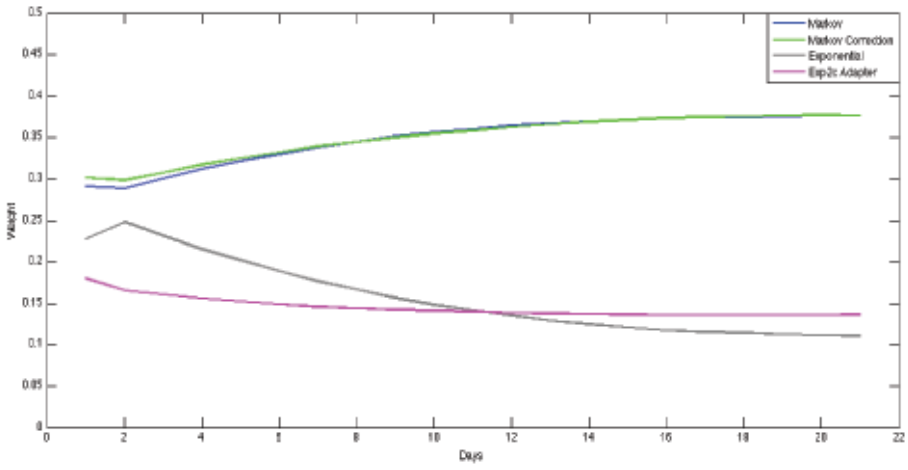

Figure 8.17: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 1.

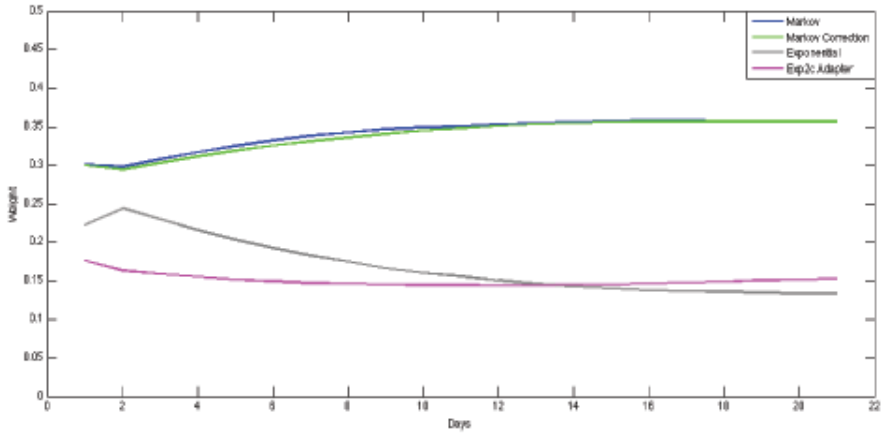

Figure 8.18: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 2.

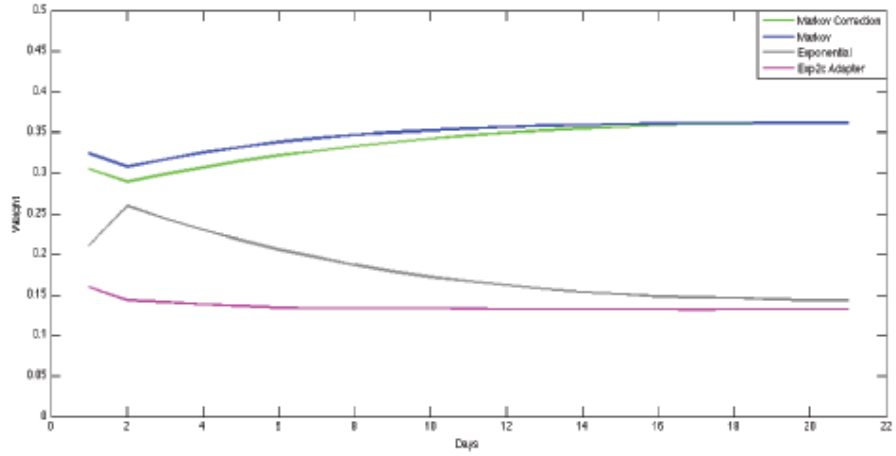

Figure 8.19: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 3.

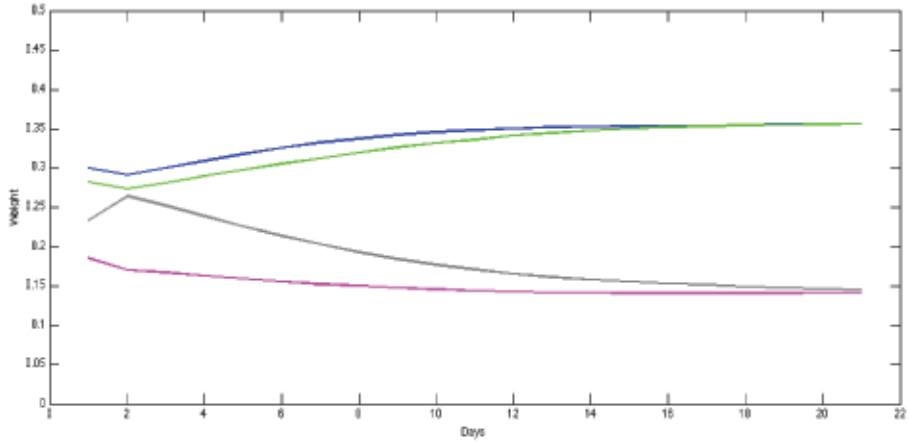

Figure 8.20: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 4.

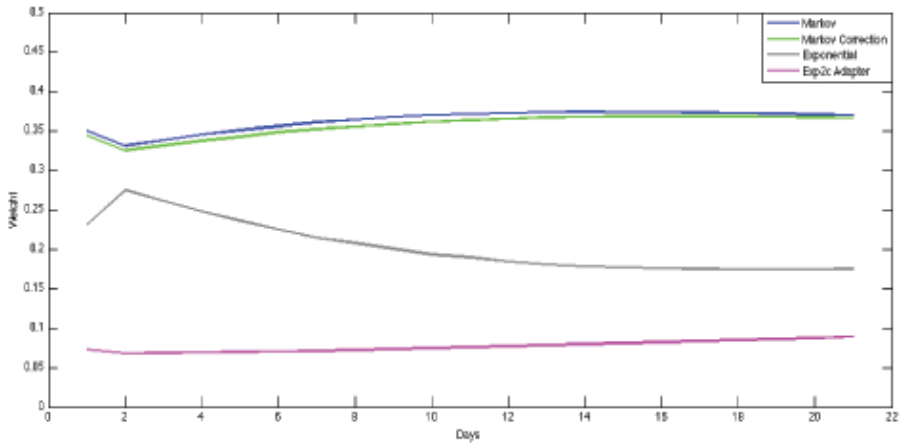

Figure 8.21: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 5.

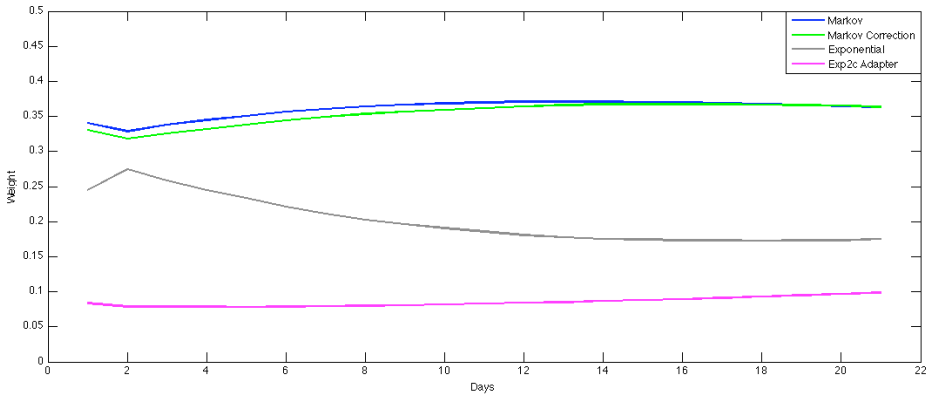

Figure 8.22: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 6.

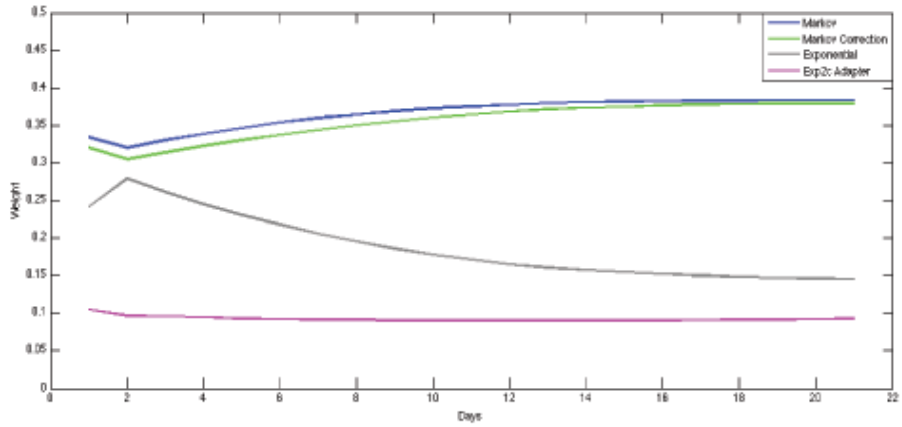

Figure 8.23: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 7.

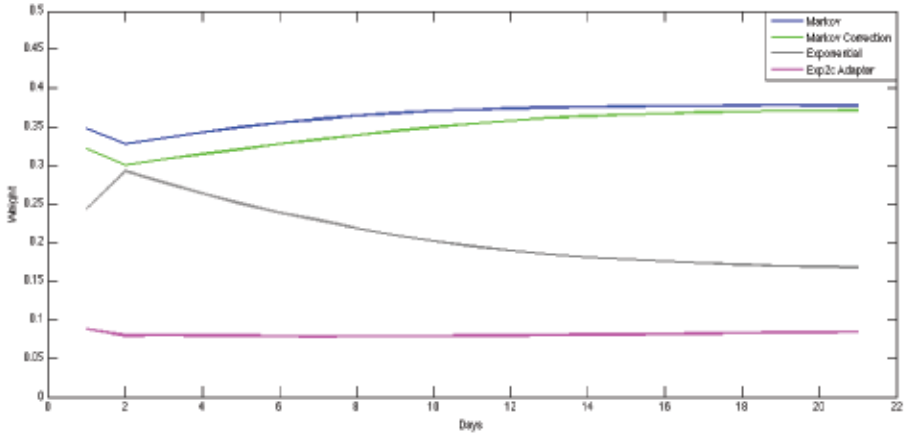

Figure 8.24: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 8.

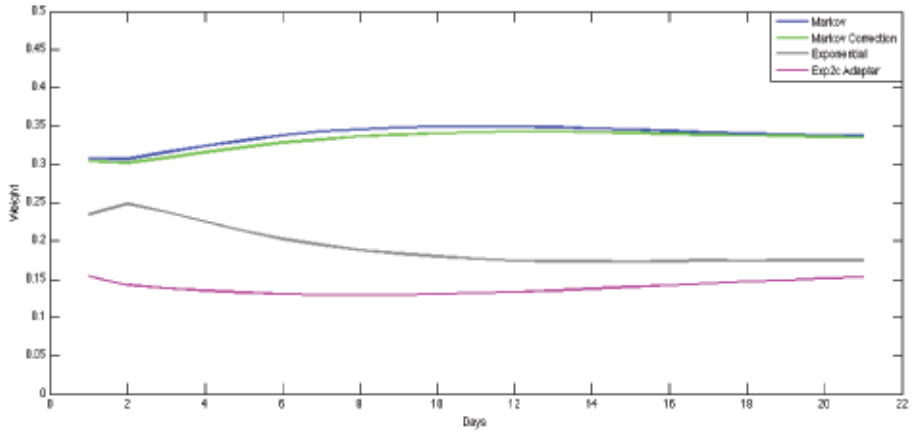

Figure 8.25: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 9.

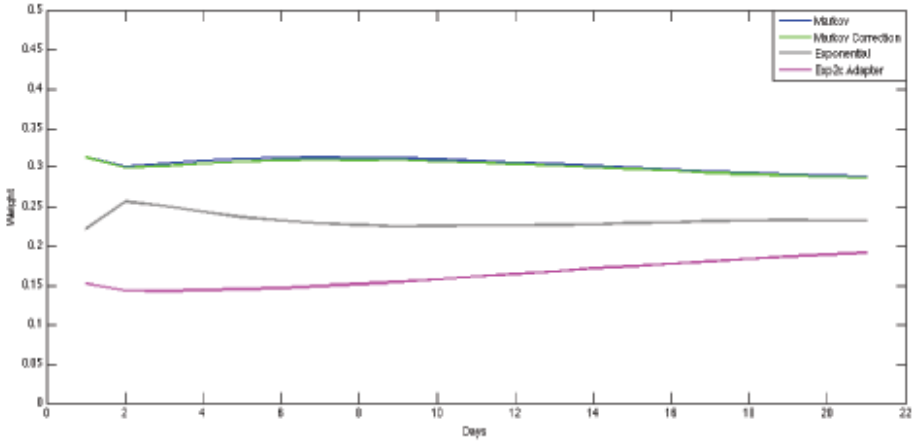

Figure 8.26: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 10.

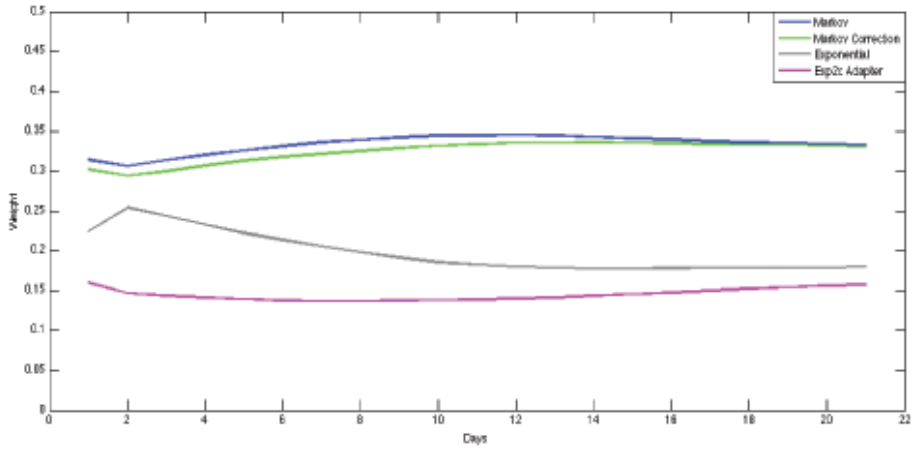

Figure 8.27: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 11.

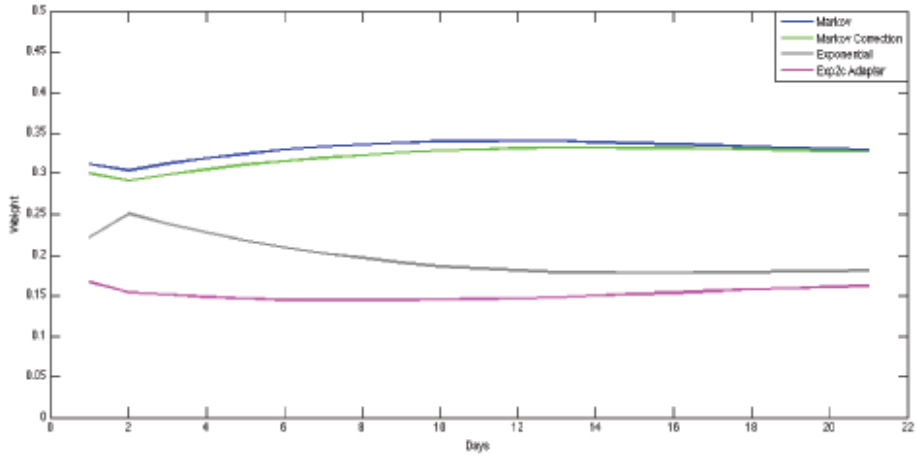

Figure 8.28: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 12.

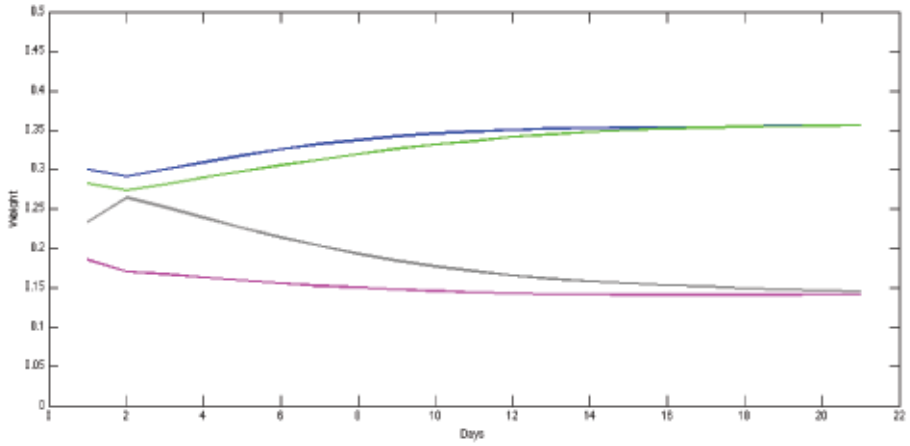

Figure 8.29: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 13.

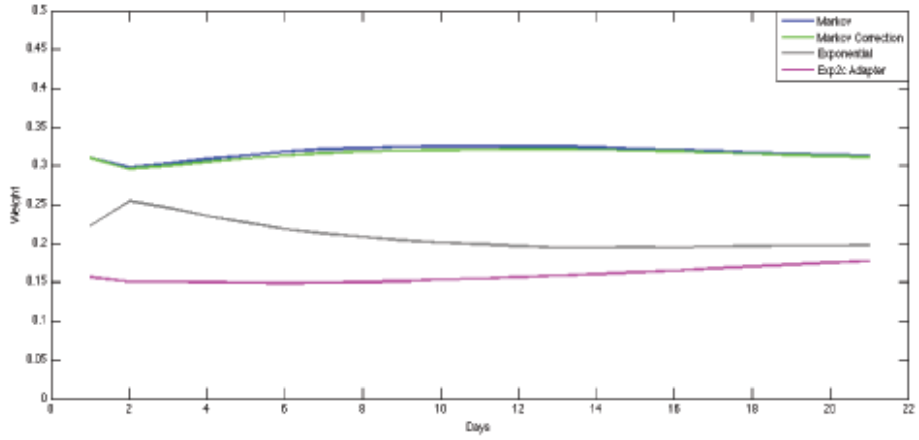

Figure 8.30: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 14.

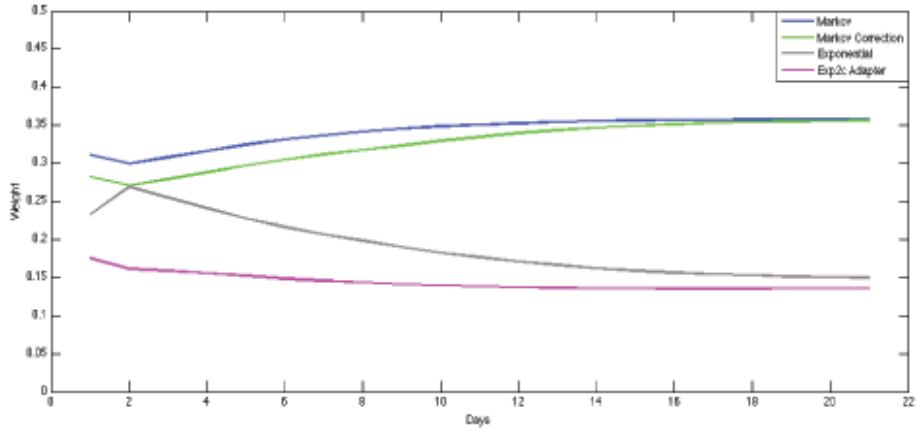

Figure 8.31: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 15.

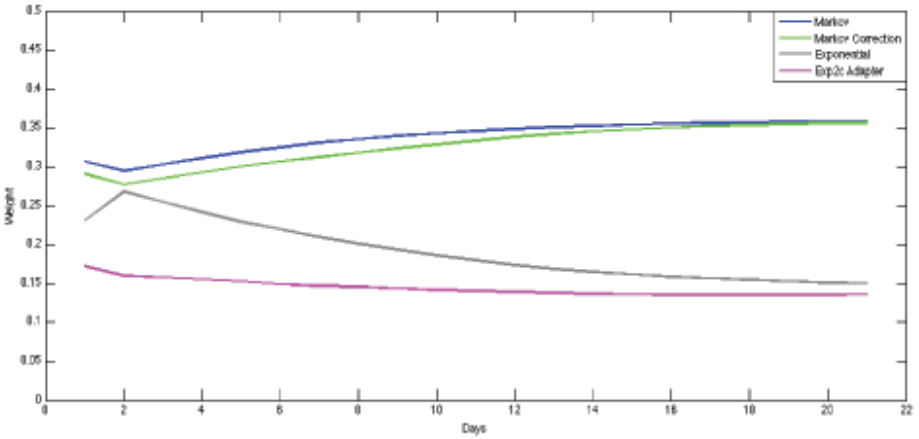

Figure 8.32: The weights for the Markov, Markov-correction, Exponential and Exponential 2c adapter calculated with the exponential variance method with a *tau* of 5 for product 16.

In this part of the tac.xconf file the variance calculation method can be set to "block" of "exponential" with respectively its horizon or tau value:r

```
<!-- SALES setup -->

      <selector
   class="edu.umn.cs.tac.oracle.eval.ModelSelectionConfiguredObject"
```

```
name="model-selection">
<parameters
  planning-horizon = "20"
  block-weight-horizon = "10"
  first-day-of-calculations = "2"
  number-of-segments = "16"
  days-in-game = "220"
  tau = "1.0"
  variance-calculation-method = "block"
  smoothed-min-max-price = "smoothed-min-max-price"/>
  <predictors name = "exponential-smoother-regime-prediction"/>
  <predictors name = "markov-prediction"/>
  <predictors name = "markov-correction-prediction"/>
  <predictors name = "exps2c-adapter"/>
  <order-pricers name = "probability-of-acceptance"/>
  <order-pricers name = "probability-of-acceptance"/>
  <order-pricers name = "probability-of-acceptance"/>
  <order-pricers name = "probability-of-acceptance"/>
   </selector>
```

This is the part of the tac.xconf file where the transition method with its alpha and the transition-length can be changed:

```
<evaluator
       class="edu.umn.cs.tac.oracle.eval.RegimeTrendCombiner"
       name="markov-prediction">
       <source name="markov-prediction-0" start="0"/>
       <source name="markov-prediction-30" start="25"/>
       <source name="markov-prediction-200" start="195"/>
       <parameters
       transition-length="10"
       method="sigmoid"
       alpha=1.0/>
      </evaluator>

      <evaluator
       class="edu.umn.cs.tac.oracle.eval.RegimeTrendCombiner"
       name="markov-prediction">
       <source name="markov-prediction-0" start="0"/>
       <source name="markov-prediction-30" start="25"/>
       <source name="markov-prediction-200" start="195"/>
       <parameters
       transition-length="10"
       method="sigmoid"
       alpha=1.0/>
      </evaluator>
```

```
<evaluator
 class="edu.umn.cs.tac.oracle.eval.RegimeTrendCombiner"
 name="markov-correction-prediction">
 <source name="markov-correction-prediction-0" start="0"/>
 <source name="markov-correction-prediction-30" start="25"/>
 <source name="markov-correction-prediction-200" start="195"/>
 <parameters
 transition-length="10"
 method="sigmoid"
 alpha=1.0/>
</evaluator>
```